NAVAL POSTGRADUATE SCHOOL

MONTEREY, CALIFORNIA

THESIS

CHARACTERIZING CROWD PARTICIPATION AND PRODUCTIVITY OF FOLDIT THROUGH WEB SCRAPING

by

Jonathan A. Yee

March 2016

Thesis Advisor: Geoffrey Xie
Second Reader: Justin P. Rohrer

Approved for public release; distribution is unlimited

THIS PAGE INTENTIONALLY LEFT BLANK

REPORT DOCUMENTATION PAGE			*Form Approved OMB No. 0704–0188*
Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instruction, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188) Washington, DC 20503.			
1. AGENCY USE ONLY ***(Leave blank)***	**2. REPORT DATE** March 2016	**3. REPORT TYPE AND DATES COVERED** Master's thesis	
4. TITLE AND SUBTITLE CHARACTERIZING CROWD PARTICIPATION AND PRODUCTIVITY OF FOLDIT THROUGH WEB SCRAPING			**5. FUNDING NUMBERS**
6. AUTHOR(S) Jonathan A. Yee			
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey, CA 93943-5000			**8. PERFORMING ORGANIZATION REPORT NUMBER**
9. SPONSORING /MONITORING AGENCY NAME(S) AND ADDRESS(ES) N/A			**10. SPONSORING / MONITORING AGENCY REPORT NUMBER**
11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government. IRB Protocol number ____N/A____.			
12a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release; distribution is unlimited			**12b. DISTRIBUTION CODE**

13. ABSTRACT (maximum 200 words)

Citizen science, scientific work done by non-experts, is an emerging method of continuing scientific investigation. In recent years, Crowdsourced Science Games (CSSGs) have become a particular area of research. In this model, citizen scientists play a video game in order to help solve scientifically hard problem sets. Recent work has shown CSSGs are severely affected by low engagement rates (ER) and a disproportionate amount of work done by a small subset of the entire player base. In this thesis, we will examine Foldit, a seemingly successful CSSG. In the absence of publicly available data, we used web scraping to obtain data on a daily basis from a player scoreboard from June 1, 2015, to February 15, 2016, and from an accumulated puzzle database encompassing the lifetime of Foldit. Utilizing previous methodology quantifying the productivity of CSSGs, we show that Foldit continues to draw players despite a gradually declining number of active users. Furthermore, a core base of experienced players contributes the most to the game. With these two factors, Foldit's game design and emphasis toward creating a small but highly trained player subset provide a strong argument for a more productive CSSG over a more entertainment-focused, casual style of game.

14. SUBJECT TERMS crowdsourcing, Foldit, whale effect, game analytics, data analytics, crowdsourced serious games, web scraping			**15. NUMBER OF PAGES** 99
			16. PRICE CODE
17. SECURITY CLASSIFICATION OF REPORT Unclassified	**18. SECURITY CLASSIFICATION OF THIS PAGE** Unclassified	**19. SECURITY CLASSIFICATION OF ABSTRACT** Unclassified	**20. LIMITATION OF ABSTRACT** UU

NSN 7540–01-280-5500 Standard Form 298 (Rev. 2–89)
Prescribed by ANSI Std. 239–18

THIS PAGE INTENTIONALLY LEFT BLANK

Approved for public release; distribution is unlimited

CHARACTERIZING CROWD PARTICIPATION AND PRODUCTIVITY OF FOLDIT THROUGH WEB SCRAPING

Jonathan A. Yee
Civilian, Scholarship for Service
B.S., University of California, San Diego, 2005

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN COMPUTER SCIENCE

from the

NAVAL POSTGRADUATE SCHOOL
March 2016

Approved by: Geoffrey Xie
Thesis Advisor

Justin P. Rohrer
Second Reader

Peter Denning
Chair, Department of Computer Science

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

Citizen science, scientific work done by non-experts, is an emerging method of continuing scientific investigation. In recent years, Crowdsourced Science Games (CSSGs) have become a particular area of research. In this model, citizen scientists play a video game in order to help solve scientifically hard problem sets. Recent work has shown CSSGs are severely affected by low engagement rates (ER) and a disproportionate amount of work done by a small subset of the entire player base. In this thesis, we will examine Foldit, a seemingly successful CSSG. In the absence of publicly available data, we used web scraping to obtain data on a daily basis from a player scoreboard from June 1, 2015, to February 15, 2016, and from an accumulated puzzle database encompassing the lifetime of Foldit. Utilizing previous methodology quantifying the productivity of CSSGs, we show that Foldit continues to draw players despite a gradually declining number of active users. Furthermore, a core base of experienced players contributes the most to the game. With these two factors, Foldit's game design and emphasis toward creating a small but highly trained player subset provide a strong argument for a more productive CSSG over a more entertainment-focused, casual style of game.

THIS PAGE INTENTIONALLY LEFT BLANK

TABLE OF CONTENTS

I. INTRODUCTION....1
- **A. MOTIVATION....1**
- **B. PROBLEM STATEMENT....2**
- **C. RESEARCH QUESTIONS....3**
- **D. METHODOLOGY....3**
- **E. SCOPE AND LIMITATIONS....4**

II. BACKGROUND....5
- **A. INTRODUCTION....5**
- **B. HISTORY OF VOLUNTEER COMPUTING....5**
- **C. CROWDSOURCING....6**
- **D. THE POWER OF COLLABORATION....7**
- **E. CROWDSOURCING AND ALTRUISTIC SCIENCE RESEARCH....8**
- **F. CROWD SOURCED SERIOUS GAMES....10**
  - **1. Verigames....10**
    - ***a. Solution 1: Circuitbot and Dynamakr....10***
    - ***b. Solution 2: Flow Jam and Paradox....11***
    - ***c. Solution 3: Ghost Map and Hyperspace....12***
    - ***d. Solution 4: StormBound and Monster Proof....12***
    - ***e. Solution 5: Xylem and Binary Fission....13***
  - **2. Foldit....13**
    - ***a. Biochemical Discovery....14***
    - ***b. Gameplay....15***
    - ***c. Social Aspects and Rewards....16***
    - ***d. Results and Published Papers....16***
    - ***e. Other Biochemical Games....17***
- **G. GAME ANALYTICS....17**
  - **1. Product Life Cycle....17**
  - **2. Active Users....20**
  - **3. The "Whale Effect"....20**
  - **4. Surveys....21**
- **H. SUMMARY....21**

III. METHODOLOGY....23
- **A. INTRODUCTION....23**
- **B. DATA METRICS....23**

1. Whale Effect Graph 24
2. Engagement Rate 26
C. FOLDIT WEBSITE 27
D. WEB SCRAPING 28
1. Scraping Code 29
2. BeautifulSoup 30
E. DATA COLLECTION 31
F. INFORMATION PARSING 32
G. DATA VERIFICATION 34
H. SUMMARY 34

IV. ANALYSIS AND EVALUATION 35
A. INTRODUCTION 35
B. LIMITATIONS 35
C. USER DATA 36
D. CURRENT USER PARTICIPATION LEVELS 40
E. TASK ENGAGEMENT RATE 43
F. WHALE EFFECT GRAPH RESULTS 45
G. SOCIAL INTERACTIONS, GAME DESIGN, AND PLAYER RETENTION 49
H. SUMMARY 50

V. CONCLUSION AND FUTURE WORK 51
A. SUMMARY OF WORK 51
B. FUTURE RESEARCH 51

APPENDIX A. PYTHON CODE FOR SOLOIST SCRAPER 53

APPENDIX B. PYTHON CODE FOR PUZZLE SCRAPER 57

APPENDIX C. PYTHON CODE FOR CSV COMPARISON TOOL 61

APPENDIX D. PYTHON CODE FOR CSV COMBINING TOOL 63

APPENDIX E. PUZZLE DATA PER MONTH—JUNE 2015-FEB 2016 65

LIST OF REFERENCES 75

INITIAL DISTRIBUTION LIST 81

LIST OF FIGURES

Figure 1. Different Stages in a Product Life Cycle 18
Figure 2. Evolution of Player Skill Level 19
Figure 3. Example of a Foldit Puzzle Scoreboard 25
Figure 4. Whale Effect Graph (WEG) 26
Figure 5. Foldit Soloist Leaderboard 28
Figure 6. CDF of Puzzle Attempts—May 2008-February 2016 38
Figure 7. New Users Added Shown as a Portion of Total Users 41
Figure 8. CDF of Puzzle Attempts—June 2015–February 2016 42
Figure 9. WEG of New Users and Old Users 43
Figure 10. Verigames WEG 47
Figure 11. Lorenz Graph of Other CSSGs 47
Figure 12. WEG from Cumulative Foldit Data 48
Figure 13. Foldit WEG divided into yearly time periods 48

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF TABLES

Table 1. Cumulative FoldIt User Data—May 2008 through February 2016 37
Table 2. Foldit User Data by Year 39
Table 3. New and Returning User Data—June 2015-February 2016 42
Table 4. TER of Foldit by Month 44
Table 5. Foldit Puzzle Attempts by AU per Month 45
Table 6. June 2015 Puzzle Data 65
Table 7. July 2015 Puzzle Data 66
Table 8. August 2015 Puzzle Data 67
Table 9. September 2015 Puzzle Data 68
Table 10. October 2015 Puzzle Data 69
Table 11. November 2015 Puzzle Data 70
Table 12. December 2015 Puzzle Data 71
Table 13. January 2016 Puzzle Data 72
Table 14. February 2016 Puzzle Data 73

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF ACRONYMS AND ABBREVIATIONS

ARPU	Average Return Per User
AU	Active Users
BOINC	Berkeley Open Infrastructure for Network Computing
CDF	Cumulative Distribution Function
CPU	Central Processing Unit
CSSG	Crowdsourced Serious Game
CSV	Comma Separated Value
DARPA	Defense Advanced Research Projects Agency
DAU	Daily Active Users
ER	Engagement Rate
F2P	Free-to-Play
GIMPS	Great Internet Mersenne Prime Search
GPU	Graphics Processing Unit
GWAP	Games with a purpose
IRC	Internet Relay Chat
MAU	Monthly Active Users
mTER	Monthly Task Engagement Rate
PAU	Puzzle Active Users
PC	Personal Computer
PvE	Player-versus-Environment
PvP	Player-versus-Player
TER	Task Engagement Rate

THIS PAGE INTENTIONALLY LEFT BLANK

ACKNOWLEDGMENTS

I would like to thank my thesis advisor, Professor Geoff Xie, for the endless support, insightful comments, and intense discussions. Throughout this process, I have learned that there is more to a thesis than just writing and research. Just like anything in life, one must be challenged to work harder, think differently, and persevere in the face of adversity.

I would like to thank my family, especially my mother and father. They have given me a roof over my head and supported every decision I have made during this long journey. My sisters, Aja and Portia, have given me the encouragement and feedback to keep on chugging along.

Likewise, I would not be here without my friends Gerard, Fadia, and Tim; thank you for being there to let me vent when things were not going my way. In addition, I want to thank the numerous friends I have made from Twitch and other online forums. During this process, I have learned that gaming is not just about entertainment but also about community and social interaction.

Lastly, it has also been my pleasure to make some great friends in my classmates from Cohort XV: Henry (honorary), Stephen, Leslie, and Sean. Somehow, we made it through the Great Filter and emerged as graduates. I am just as shocked as you are.

THIS PAGE INTENTIONALLY LEFT BLANK

I. INTRODUCTION

A. MOTIVATION

Ever since the personal computer (PC) revolution began in the early 1980s, the pervasiveness of individual computer ownership has continued to increase. In 2015, over 300 million PCs were sold in a year [1]. Likewise, computing power has continued to follow Moore's Law, the principle that the numbers of transistors on an integrated circuit will double every two years, with the total number of transistors produced reaching astronomical levels [2]. Currently, a top-of-the-line, commercially available central processing unit (CPU) has four cores, eight threads, and a clock rate up to 4.0 GHz. Microarchitecture abilities are so advanced that the same CPU can house a graphic processing unit (GPU) comparable to a low-end dedicated graphics card [3]. With access to PCs running powerful CPUs, the average person has an extraordinary amount of computing resources.

Researchers have taken notice of this phenomenon and have developed several ways to harness this power. One way is to have individuals donate their computer's processing power in order to do work. This method, called volunteer computing, allows a program to utilize idle computer time to work on computationally difficult problems [4]. Aside from installing the requisite computing software, the individuals donating their computer power do not interact with the computing process. Indeed, this type of computing relies upon working in the background [5]. The software, including downloading and uploading work units, completes any computation without any user input. While a single PC cannot possibly replicate the performance capabilities of a supercomputer, the combined distributed power of many computers at once can create a similar capacity. According to Anderson [6], principal investigator for the Berkeley Open Infrastructure for Network Computing (BOINC), "about 900,000 computers are actively participating in volunteer computing. Together they supply about 10 PetaFLOPS (trillion floating-point operations per second) of computing power." At the time of this writing, the fastest super computer, Tianhe-2, in comparison, can perform 33.86 PetaFLOPS [7].

Still, humans are better than computers at completing some tasks. While computers are superior at prolonged computation, they lack the creativity and spatial acuity innate to humans. In particular, humans are still vastly superior to computers in pattern and object recognition. Of the 14 most common models used in computer vision, none was able to replicate human capabilities [8].

While both software verification and protein discovery can be automated to a degree, computer processing alone cannot fully accomplish the task [9, 10]. As a result, a human expert is needed to analyze the output. From this realization, a new genre of distributed computing unfolded. Called crowdsourced serious games (CSSGs), this genre combines volunteer computing and active human participation in order to solve complex problems [11]. The two main programs under analysis in this thesis, Verigames and Foldit, use CSSG theory in order to perform software formal verification and protein discovery, respectively. It is the hope of CSSG designers that harnessing the superior spatial skills of humans will help reduce the work of experts. Decreasing the workload not only helps reduce cost, but also allows an expert to avoid spending time on trivial tasks.

B. PROBLEM STATEMENT

Unlike traditional volunteer computing, where the user does not interact with the software, CSSGs require active participation in order to generate results. As such, user retention and user quality are of utmost importance. In the first round of Verigames development, initial player interest peaked at the launch of game. Within six months, interest for all games fell to near zero [12]. In contrast, a limited survey of Foldit users over a two-month period in 2012 estimated a current player base of 200–300 players despite being a 4-year-old game at the time of sampling [13]. Despite the figures from the aforementioned study, active user statistics were not gathered in a quantitative manner. Instead, the authors estimated the active user base from their own observations. Additionally, the purpose of their study was to create a survey to examine a user's motivation for playing a CSSG, not an analysis of user retention.

As of today, the current Foldit player environment has not been studied intimately. The purpose of this thesis is to implement commonly used game analytics and apply them to Foldit to determine whether the game continues to attract and retain players. From these results, we should then be able to compare if Foldit follows similar trends to other CSSGs. If the trends differ, we can then make suggestions on how to improve other CSSGs, namely, other games in development under the Verigames project.

C. RESEARCH QUESTIONS

Foldit has been relatively unaltered during its seven-year lifespan. During this time, it has been the subject of seven significant papers, one of which demonstrated that a candidate protein created by Foldit players contributed to the elucidation of a HIV enzyme [14, 15]. Given the scientific impact and relatively high number of active users, the Foldit framework can be applied to other CSSG projects, namely, Verigames, in order to attract new players and retain current players.

In an ideal situation, a large percentage of players would continue to play the game. Likewise, the contribution from each player base percentile is proportionate to its size. Previous work analyzing two anonymous Verigames games saw a relatively low engagement rate (ER) and a strong whale effect [16]. Such metrics have not been assessed on a successful CSSG like Foldit.

D. METHODOLOGY

The research in thesis is dependent on collecting quantitative data through web scraping and further analysis through statistical methods. Using web scraping, one can harvest any information posted on a public facing website. This includes images, hyperlinks, and text. In this case, scraping was a necessity since we were denied access to raw data by the developers. However, Foldit had a public facing web portal with a myriad of useful data including total number of users and individual player scores. To facilitate web scraping, we wrote a custom scraping program. After data collection, information was analyzed using statistical methods common to the game industry. These include approximations for player engagement and user contributions. Finally, comparisons were made in accordance with previous studies conducted against a collection of CSSGs

collectively sponsored by Defense Advanced Research Agency's (DARPA) Verigames project [16].

E. SCOPE AND LIMITATIONS

Without direct access to the developer's database, we were constrained to data found on Foldit's public facing website. While this data is presumably the same, it is unknown if it has been sanitized in some way. We were also constrained by timing during our data collection. Since we could not collect data from the start of the Foldit project, our data lacks historical depth. As a result, presumptions about Foldit can be applied only to the current posture of the project.

II. BACKGROUND

A. INTRODUCTION

In this chapter, we provide a brief overview of previous attempts of using crowdsourcing in gaming. In order to understand the potential benefits of crowdsourcing, one must first understand gaming industry terms, the history of crowdsourcing, and the development of CSSGs. From this background, we will demonstrate why crowdsourcing and distributed computing show such great promise in solving difficult computing problems.

B. HISTORY OF VOLUNTEER COMPUTING

George Woltman founded the first major volunteer computing project in 1996 [17]. This project, called the Great Internet Mersenne Prime Search (GIMPS), implemented freely available software in order to compute high-order Mersenne primes. Calculating high-order primes is computationally intensive. The largest known prime, the Mersenne prime $2^{74,207,281} - 1$, took a single computer 30 days to calculate. However, behind that single computer, it took thousands of other computers sifting through millions of non-candidate numbers to find a real Mersenne prime [18]. The GIMPS developers recognized that most computer systems are often idle, running without any program to process aside from system processes. The developers believed they could leverage wasted spare computing cycles to tackle this problem in a distributed manner. In this method, one divides a large problem into smaller, easier to compute data blocks. Individual computers then calculate these smaller work units and report to a centralized database. Without a true distributed system existing, users initially had to email the developers for problem sets to compute. Nevertheless, within the first few weeks of the project being instantiated, the 35th Mersenne prime was discovered. Further demonstrating the power of distributed computing, GIMPS has discovered every new Mersenne prime since its inception [17].

As other researchers began to take notice of the power of distributed computing, several distributed projects appeared in order to take advantage of both the publicity and

capabilities of this new medium. One of the first to take advantage of the newfound popularity of volunteering computing was SETI@Home [19]. This project was designed to use spare CPU cycles to scan astronomical radio data for abnormal power spikes indicative of extraterrestrial life. From this project, a software-based distributed computing platform called the Berkeley Open Infrastructure for Network Computing (BOINC) was born. This platform was developed as a framework for other large-scale distributed computing projects. Current projects such as Rosetta@home, Malaria Control Project, and Einstein@Home use this framework to compute everything from protein structures to malarial vector modeling to gravitational wave detection. All told, more than 3 million people have signed up to use their idle computer cycles as a way to further scientific progress [20].

C. CROWDSOURCING

As the idea of volunteer computing continued to evolve, other technologists envisioned harnessing the burgeoning number of home computer users as a method to tackle other problems. Howe first coined "Crowdsourcing" in an article for *Wired* magazine titled "The Rise of Crowdsourcing" [21]. In the article, Howe used the same idea of distributed computing to describe what he called "distributed labor." Traditional labor practices involve hiring a single person or a group to accomplish a job by a certain time with an agreed upon payment for accomplishing the task. At the same time, with the rise of globalization, a new labor paradigm came into fruition. This model, called outsourcing relied upon using cheaper, foreign labor in order to do work at a fraction of the cost of a local worker. Combining the terminology of distributed computing and outsourcing, Howe created this definition:

> Simply defined, crowdsourcing represents the act of a company or institution taking a function once performed by employees and outsourcing it to an undefined (and generally large) network of people in the form of an open call. This can take the form of peer-production (when the job is performed collaboratively), but is also often undertaken by sole individuals. The crucial prerequisite is the use of the open call format and the large network of potential laborers. [21]

From this definition, crowdsourcing relies upon both collaboration and competition to accomplish a goal. Crowdsourcing as a business model acts to lower costs

by having an open bidding process and accomplish the goal by dividing the collaboration into smaller units.

Brabham [22] further refined the definition of crowdsourcing by stating, “All crowdsourcing applications consist of an organization that issues a task to an open online community, and the community participates in accomplishing the task for the benefit of the organization.” This is to differentiate crowdsourcing from commons-based peer production. In commons-based peer production, the organizational hierarchy tends to be less rigid with an emphasis put upon group collaboration toward a common goal. Some current examples of this type of production are Linux, GNU, and *Wikipedia*. In contrast, crowdsourcing can be chaotic, with participants having less of an incentive to complete a task.

D. THE POWER OF COLLABORATION

In recent gaming history, multiuser game collaboration has become a source of both amusement for players and an active social experiment. One of the more notable collaborative efforts took place on the website Twitch. Twitch, the world’s largest gaming website and one of the most active websites on the Internet, generates millions of views a day and has thousands of broadcasters playing a variety of games [23]. On Twitch, anyone with a webcam and an Internet connection can create a channel in order to broadcast their gameplay. Unlike static video sites like YouTube, Twitch broadcasts are live. One essential component to a streaming channel is broadcaster/viewer interaction. To facilitate this communication, Twitch channels have an embedded Internet Relay Chat (IRC) channel so viewers can type and the broadcaster can respond. IRC is not a new technology. As such, many programs have been written so that typed responses can be read, and the program will have a scripted response. In general, this type of program is called a bot.

Designed as a social experiment to test social collaboration, the bot “TwitchPlays” parsed incoming text for strings designating commands [24]. The bot then made an action corresponding to the command in the game. Given the frenetic nature of thousands of players simultaneously inputting commands, the programmers for

TwitchPlays implemented two primary modes of game interaction: anarchy and democracy mode. Anarchy mode periodically polled user inputs and randomly chose a command during a given window of time. Democracy mode polled all user inputs and grouped them accordingly. The most frequently chosen command was then executed.

In order to test the bot, the programmers decided to play Pokémon, a turn-based game with only simple inputs and straightforward gameplay [24]. With the numbers of players participating constantly in flux, commands being issued without formal collaboration, lag time between input and screen action, and intervention from bad actors actively seeking to disrupt the game, it was speculated that the game might never be beaten. However, after a few days, the player actions moved toward beating the game. After 16 days of real-time gameplay and over 122 million commands given, the game community managed to beat the game [25].

E. CROWDSOURCING AND ALTRUISTIC SCIENCE RESEARCH

In computer science, we often employ various methods to speed up computation. Modern processors have multiple cores, programs are designed with multi-threading, and parallel processes distribute work for an application. Recognizing the similarities between these principles and the billions of hours spent by users playing video games, von Ahn, developed the concept of "games with a purpose" (GWAP) [26]. According to von Ahn, a GWAP is a class of game "in which people, as a side effect of playing, perform tasks computers are unable to perform" [26]. Despite tremendous advancements in computer processing, humans are still better at certain tasks then computers. One example of this is image labeling. Although programs have improved in automating picture identification, proper machine learning and artificial intelligence still lag behind human visual cues. Humans, for example, are better than computers at describing attributes and identifying multiple objects in a picture [27].

Taking advantage of the inherent strengths of human visual and pattern recognition, von Ahn and Dabish created the ESP Game" [27]. In this game, two players were matched. The two players were presented with an image. Players then chose a word they felt best described the image. Whenever the two players agreed on a word, an

indicator would increase. Once a certain threshold of agreed upon words was achieved, another image would appear. During post-processing of data, von Ahn and Dabish found that the most commonly agreed upon words for an image often correlated with similar, yet objectively different images. As such, programs that are good at detecting broad patterns can have these rules integrated into their labeling guidelines. Overall, the user contribution was quite significant. More than 13,000 players participated during a four-month period and contributed over a million labels for 300,000 images. Internal statistics also revealed that more than 80% of users returned to play the game. The authors took this to indicate that the players enjoyed the game enough to keep on playing.

Many studies have been done on the motivation for playing video games [28]. In a traditional commercial game, game modes can be broadly categorized into Player versus Environment (PvE) and Player versus Player (PvP) games. In a PvE game style, the player interacts primarily with statically generated goals, items, puzzles, quests, and other game-related material. Interaction with other players is at a minimum, although cooperative play is often integrated to enhance rewards and provide an environment in which some social activities can take place. Competition can exist in these games by ranking player skills using a scoreboard. On the other hand, PvP games are developed around the idea of competition between players [29]. This style of game is best represented by first-person shooter games, where interaction with computer-generated objectives is minimized, and, instead, the progress of the game is propelled by defeating the opponent.

Although there have been several strategies to entice players to keep on playing a CSSG, there has not been a definitive study on the impact that these different play styles have had on user retention. Instead, most studies have focused upon user motivation, program usability, and scientific results [13, 28, 30, 31]. In these surveys, altruism, scientific interest, and community involvement have all ranked high as reasons for playing a particular game.

While not directly competitive, the purpose of a CSSG is to complete a scientific goal, using a scoreboard, rankings, and teams, elements that mirror many of the characteristics of traditional games.

F. CROWD SOURCED SERIOUS GAMES

By its very nature, the outcome of a science game cannot be known in advance. If a scientific problem is trivial, there is no benefit in using crowd sourced computing. However, this creates an interesting conundrum: How does one create a game that is fun and interesting but also serves the purposes of science? This question was a central tenet behind two CSSG projects: Verigames and Foldit.

1. Verigames

With continued advances in computer technology, many critical functions are leaving the realm of human control and being entrusted to software. However, a common axiom in computer science is that if it is software, there will be bugs. In order to be assured that a piece of software is completely free of bugs one must conduct a complete formal verification. In 2009, Klein et al. produced a paper detailing the process of completely verifying the seL4 microkernel. This kernel was composed of 8,700 lines of C code and 600 lines of assembly. Using custom built tools and formal proofs, verification took 20 combined person-years [32]. Given that a modern operating system is composed of millions of lines of code, it is easy to imagine the difficulty of this problem.

In response to this critical need, the Defense Advanced Research Agency (DARPA) initiated the Verigames project. This project acted as a venue to explore if CSSGs could be developed in order to verify code. DARPA initially supported several institutions that independently produced CSSGs in their own manner. In total, five solutions were developed and evaluated for successfulness.

a. Solution 1: Circuitbot and Dynamakr

Circuitbot and Dynamakr were designed to verify C-language programs using pointer analysis [33]. Using the games as a front-end, the developers hoped the players would produce so-called “points-to-graph” in which a specific point was an extrapolation of a memory location. Each time a player created a point, a connection would be made on a graph. These connections, called arcs, acted as software constraints to be analyzed

using CodeHawk, a static analysis program. Finally, CodeHawk would then be able to detect if memory overruns occurred.

The developers first created Circuitbot, a turn-based strategy game [33]. In this game, a player would deploy exploration vehicles to planets and harvest resources. Game advancement relied upon sufficient facilities being built on a planet. In actuality, this game strategy steered players toward forming arcs. Unfortunately, the developers realized that worker production was too small and verified little code.

In their second attempt, the developers created Dynamakr [33]. Instead of a turn-based strategy, Dynamakr was a dynamically driven game. A player would connect and find patterns in order to launch the actual game engine. The number of patterns found in a specific puzzle would determine the amount of points and abilities for the forthcoming stage. This action greatly expanded the number of initial connections made by the player, and encouraged further gameplay by rewarding the player for increased work.

b. Solution 2: Flow Jam and Paradox

Flow Jam and Paradox used type theory in order to verify software [34]. In programming, assignment statements are read from left to right. With this logic, variables on the left of an assignment are a super type of those on the right. This naturally provides for a constraint system. If a player successfully solves a problem, the extrapolated code can be inferred to be free of error. If the puzzle proves intractable, then an expert can review the piece of code.

The first iteration of this game was Flow Jam [34]. In this game, the player was presented with a series of blocks interconnected with pipes. Each block would have an assigned value. The player would then open a “valve” on a block for a specific pipe in order to change the value. The ultimate goal was to reach an assigned for the puzzle. Unfortunately, the complexities of software verification also lead to intricate and complex puzzles. This game proved to be confusing to the player and not scalable.

The second iteration was Paradox [34]. This game used the same type verification method, but allowed for broader adjustments. In this game, a player solved puzzles by

moving elements of a hinge-like structure. Each element changed color in accordance with the type and code being verified. The player's goal was to eliminate red colored conflicts. In actuality, this system was a frontend for automated adjustment algorithms. The players acted as intermediaries looking for patterns that these automated programs would not necessarily process.

c. Solution 3: Ghost Map and Hyperspace

Ghost Map and Hyperspace used simplified models of a program [35]. Referred to as a control flow graph, each graph correlates to the execution of a program. The developers took a series of correctness problems and mapped them into the game engine. Each puzzle represented a series of possible paths that may violate the correctness of the path.

Ghost Map consisted of a series of nodes, paths, and edges [36]. A representative problem would be displayed to the player. It would be the player's goal to cleave certain paths and remove edges in order to form the desired form. The result was related to a predetermined verification done by an automated tool. If these two matched, it was said that the program's code was free from vulnerabilities. Through this action, the player's contribution helped reduce false positives and guide experts into focusing their attention elsewhere.

The second iteration, Hyperspace, kept the same underlying playstyle and verification background. Instead, player feedback was integrated into the game with improved processing times and a clearer storyline.

d. Solution 4: StormBound and Monster Proof

Stormbound and Monster Proof took the approach of players creating code assertions [37]. As the player progressed through a puzzle, the players identified patterns between a function and its inputs or outputs. The behaviors produced by these assertions were then checked according to the proper program execution.

The game execution between the two varied greatly. Stormbound was presented as a story driven game without any math involvement. Monster Proof was a resourcing

gathering game where the math behind the verification was at the forefront. Both games also dealt with unsolvable puzzles in differing manners [37]. In Stormbound, every possible assertion must first be attempted before being deemed unsolvable. In contrast, Monster Proof allows an individual player to deem a puzzle unsolvable. This puzzle then is passed to another player, in which if the puzzle is proved solvable, the correct player gains more points. If the same puzzle is flagged as unsolvable several times, it is apportioned to an expert for review.

e. Solution 5: Xylem and Binary Fission

The development of Xylem and Binary Fission typify the difficulty in creating the correct game in order to solve science problems [38, 39]. Xylem was designed to be accessible to the casual player. Designed specifically for the iPad, the developers hoped to capture audiences most likely to play a game for very brief periods. Each puzzle was math-based with the user helping find combinations and patterns in fictitious flowers and plants. These patterns translated to loop invariant problems that then were evaluated by a backend machine. Unfortunately, the developers quickly learned that a math-based game was not attracting an audience, resulting in poor engagement.

In their second attempt, called Binary Fission, the developers decided to market toward players interested in crowd science [38]. The game engine was redesigned such that player actions helped steer an automated verification system instead of producing invariants for the system to test. The developers also included a chat system in order to foster community.

2. Foldit

Foldit was based on a framework from which other CSSGs can follow. The designers of Foldit had three central tenets [40]. First, humans have exceptional abilities for spatial reasoning and 3D identification super to current computer technology. Second, one must not be a scientist in order to have the problem-solving skills useful in advancing a specific scientific domain. Finally, there must be some form of scoring system in order to promote strategies in tackling the problem, while maintaining consistency to the fundamentals of the basic science at hand.

a. Biochemical Discovery

The groundwork in Foldit was set during another volunteer computing project, Rosetta@home. Utilizing the BOINC framework, Rosetta@home used distributed computing in order to predict natural protein folding [41]. A protein is made up of a combination of amino acids. In an unaltered state, amino acids are composed in a chain, linked by a strong peptide bond. To be an active, the protein must be folded such that it properly functions. The exact details of protein folding, charged amino acids, and hydrogen bonding go beyond the scope of this thesis. Nonetheless, the lowest energy state of the protein is typically the most stable, and thus, most likely to be naturally occurring.

To accomplish this, Rosetta@home took a candidate amino acid chain and systematically tried to fold the chain into different shapes [42]. Upon each round of folding, the amount of energy used to maintain the shape was calculated and reported whenever a work packet was sent back by a volunteer machine. Depending on the energy state, the master system would either accept or reject a change in energy state. This process was repeated hundreds of times with each work packet, further refining the protein until a realistic candidate protein were formed. Since this work was purely hypothetical, it was quite possible that investigation of the protein would not have a positive result. However, the CASP7 protein, a protein responsible for apoptosis, was accurately predicted by Rosetta@home [43].

While raw computing power may seem like a solution to protein structure prediction, Rosetta@home had some sever limitations. Amino acids can be arranged in a myriad of different patterns with differing lengths of chains and types of amino acids. Rosetta@home worked in a methodical manner, avoiding making large structural changes in favor of smaller corrections [10]. This made the search space incredibly large and computationally difficult.

The Foldit developers decided to take a different approach; using the creative and spatial skills innate to humans in order to aid in protein discovery. The researchers hypothesized that humans would be able to take predictions from Rosetta and make

substantial improvements when difficult rearrangements were needed. Likewise, human competition and collaboration would allow new strategies to evolve [40].

b. Gameplay

When it comes to designing a game, developers usually look for several things. First, there must be a marketplace for the game. It can prove difficult to introduce a new game into a saturated market. Next, one must develop a game environment that is fun, yet significantly challenging enough to keep the user from growing bored. In order to test the design, a game also goes through preliminary phases of development. This allows for player feedback, bug quashing, and further refinement to the product. Finally, after hitting the market, the game should be profitable or popular enough to garner further development [44]. Yet, when it comes to designing a CSSG, the design parameters need to be altered. For one, CSSGs are scientific undertakings. Profitably is not as important as maintaining player interest. Similarly, the game must also be scientific relevant, not just fun.

Foldit was designed to be accessible to players unfamiliar with biochemistry [40]. Complex biochemical structures are simplified into easily mutable structures devoid of more advanced biochemical terminology. The game begins with target proteins being added to a database by researchers. Each week, several proteins are chosen for distribution to the Foldit community. The players then act to alter the protein by rearranging structures, forming new bonds, and stretching sheets. The game gives visual feedback whenever a player makes a change to a figure. If an unnatural move occurs, a red spiky ball shape will appear. If a sheet alteration resulted in a less favorable energy state, the threshold score will decrease. Depending on the process mode, the game will dynamically change the score for a given puzzle. Aside from aesthetics, these visual changes also reinforce the game restraints to the player. The game also takes advantage of the advanced tools imported from Rosetta. Players can use the “wiggle” tool to make automated, minute changes to the structure that would be tedious to do otherwise.

c. Social Aspects and Rewards

In designing Foldit, the developers emphasized their desire to maintain as productive of a user base as possible [40]. The believed this was best accomplished by both having players compete each other through a ranking system, and having collaboration through socialization. Unique to CSSGs in the field, Foldit encourages collaboration on projects through private messaging and forum posts. An individual user creates a profile from which all of their statistics, puzzles, rankings, and achievements are compiled. Players can also join teams and receive scores on collaborative projects.

Originally, both collaborative and soloist scores were grouped into the same scoreboard [40]. Nevertheless, in order to encourage both individual and group accomplishment, the researchers developed two separate scores, soloist and evolver. Soloist scores judge a player based solely on how their progress by themselves. Evolver scores are more complex. With an evolver-initiated puzzle, a team of players has access to the protein in question. Every player on that team can then make adjustments. Player score was then judged off their contribution to the overall final design. These profiles are publically available on the Foldit website and allow one to see where they stand in regard to their peers [45, 46].

d. Results and Published Papers

Foldit debuted in May 2008. Within five months, over 50,000 users had registered to play [19]. The initial puzzles were from known structures. After a few months of analysis, the developers realized that many of the designs created by Foldit not only matched, but at times outperformed models created in Rosetta [46]. In late 2010, the developers released a still-to-be discovered puzzle associated with a critical component of a HIV protease, an enzyme critical for HIV reproduction. A team of Foldit non-experts was able to solve the puzzle within three weeks [47, 48].

Since that time, seven significant papers have been published concerning Foldit [14]. Both the scientific results and game success have been referenced in several publications as an idealized model for CSSGs.

e. Other Biochemical Games

In April 2014, the developers of the Foldit released a new biology-based visualization game called Nanocrafter [49]. Instead of protein and molecular folding, Nanocrafter's objective was to leverage crowd ingenuity in order to create innovative DNA strands. While DNA is most commonly thought of in terms of biotechnology, DNA technology has applications ranging from self-assembling nanotechnology to logical circuits and programming.

Taking some of the lessons from their previous experience with Foldit, the developers decided to use non-objective scoring. Instead, the developers decided to implement a hybrid peer-based scoring system. Hoping to encourage novel designs, the developers believed the pressure of active competition would hinder creativity [49]. The scoring system was given to separate ratings: a practical score equating to the usefulness and viability of the molecule, and a creativity score, based off a community rating given by players themselves.

G. GAME ANALYTICS

Publishers and developers often need a way to gauge the state of the game. Was the game an initial success? Did the game see a massive user drop off after a period? Has the activity rate increased, or plateaued? How is user activity distributed across the player base? What can be done to make improvements? All of these questions can be addressed by statistically evaluating the player base.

1. Product Life Cycle

As can be seen in Figure 1, games typically follow a standard product cycle. Cook describes the evolution of a game in this manner, "Genres evolve over time as players discover, fall in love, grow bored, and then move on to other forms of entertainment" [50]. A game attracts the most interest at the beginning of its introduction. It is at this time when a game is mostly heavily promoted and the gameplay is new and fresh to the player. As a game gains traction and garners broader interest, players become more involved and spend more time playing. Eventually, both market share and player interest

reach a plateau. It is at this point when the game has its largest player base and has reached its peak revenue. Afterwards, the player base begins to shrink as players find new forms of entertainment. Finally, a few dedicated players remain through continued enjoyment, brand loyalty, or other intrinsic factors.

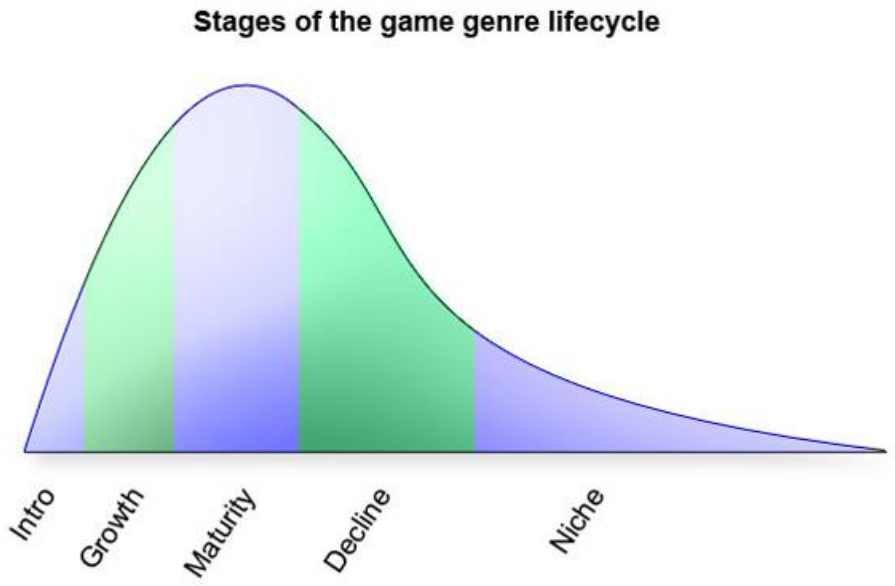

Figure 1. Different Stages in a Product Life Cycle

Source [50]: D. Cook. (2007, May 15). The circle of life: an analysis of the game product life cycle. *Gamasutra* [Online]. Available: http://www.gamasutra.com/view/feature/1453/the_circle_of_life_an_analysis_of_.php?print=1

The skills and abilities of the player base also evolve following a similar life cycle curve. Figure 2 demonstrates this life cycle. In the beginning, the gameplay is fresh and original. A novel game mechanic may enrich the experience the player, giving rise to increased interest. These players may demonstrate limited skills, but continue to play at a consistent rate. Wanting to improve, some players may invest more time and effort into improving their skill level. This leads to a subset of players with a mature skill set. These players form a core base. Not only will they play more frequently and at a higher skill level than the average player, this subset typically promotes the game to friends and family. This can lead to further player recruitment. However, it is hard for a player base

to maintain such a large core for a sustained amount of time. Eventually, the mature player base fragments or grows disinterested. What is left are lingering players composed of a small set of hardcore gamers and a larger base of infrequent, lapsed players.

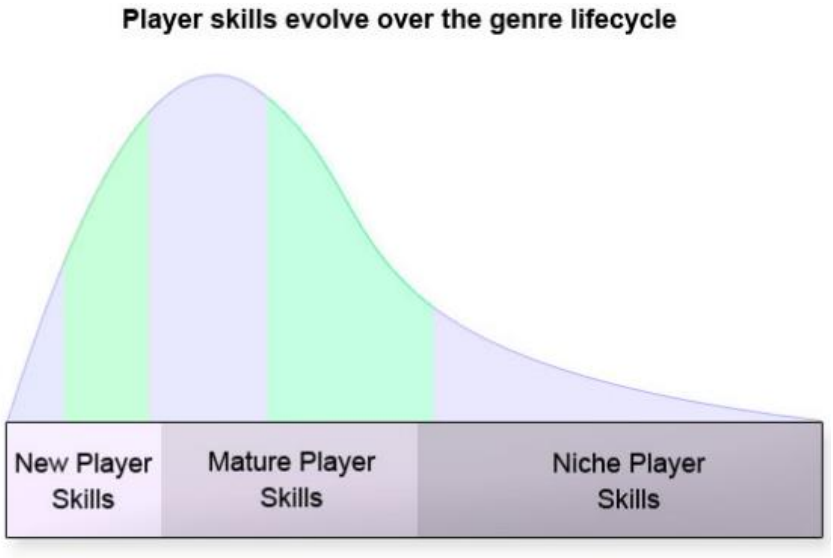

Figure 2. Evolution of Player Skill Level

Source [50]: D. Cook. (2007, May 15). The circle of life: an analysis of the game product life cycle. *Gamasutra* [Online]. Available: http://www.gamasutra.com/view/feature/1453/the_circle_of_life_an_analysis_of_.php?print=1

During each stage of a game cycle, developers must assess their position concerning future changes. One benefit of a video game, as opposed to a static product, is that a developer can provide new content. This can be accomplished by creating new challenges, adding new features, or offering new playstyles. Developers can also incentivize players by offering rewards in the form of new items or visible achievements. If a game performs poorly or continues to decline, the developers may choose to scrap any further plans and move on to new products. The ultimate decision on which way to provide can be complicated and highly reliant on user feedback, development funds, and motivation from the developer to continue to support the product.

2. Active Users

Since we are most interested in participants who actively contribute to a scientific project, it is important to use a metric that can accurately portray the user base. A traditional metric used in game analysis is the active user (AU) count. In a commercial game, the AU is usually defined as any interaction with the game [51]. The mere act of a player logging into the game could be counted as part of the AU. For commercial games is average return per user (ARPU). ARPU is best defined as the monetary value each user provides [52].

3. The "Whale Effect"

A user base for a game can be divided into three categories: minnows, dolphins, and whales. According to Nicholas Lovell, minnows spend a minimal amount, dolphins spend a moderate amount, and whales spend the most. For a commercial game, these amounts can be generalized in a $1:$5:$20 ratio from minnow to whale. In a typical free-to-play (F2P) game, one should try to achieve a 50–40-10 split between the three categories in order to be profitable and maintain an active user base [53]. Previous studies also reinforce the idea that < 5% of the user base account for the vast majority of in-game purchases [54]. Likewise, not all whales are created equal. Within the group of whales, an even smaller subset of top whales might further drive revenue for a game. Loss of this core group, while not necessarily crippling the game, may severally affect the quality of the player base and effectively negate future gains.

One outstanding question is what kind of impact does the kind of player have on total contribution? In an ideal scenario, any player of any skill level would be able to contribute. The game framework would be simple enough to be widely understood without the player being bored, overwhelmed, or confused. Likewise, the problems tasked to the players should not trivial, but at the same time understandable to a non-scientist. Brabham refers to this group as the "amateur crowd," an idealized group of hobbyists who are able to function at the same level as a single expert [22]. However, this group does not perform at an expert level. Indeed, little, if any, impact can be seen within the amateur group. Those who participate in CSSGs are often composed of a self-selected

population of highly educated individuals. Although this does not discount the benefit of CSSGs, this suggests further refinement to existing game frameworks can increase those capable of contributing, albeit at a lower rate.

4. Surveys

A 2015 Foldit survey conducted by Curtis [13] polled 37 participants and asked several questions concerning demographics, motivation, and game design. Citing the high barrier for entry into actively contributing to Foldit in comparison to other CSSGs, the survey warned against potential selection bias. Furthermore, the number of survey participants could not fully capture the Foldit player pool since user participation was voluntary.

In total, the survey found 70% of respondents had at least an undergraduate degree with over 90% of respondents having a degree in the STEM field. The highest proportion of players worked in the computer or IT industry and over 50% had participated in volunteer computing projects or other CSSGs. Those who responded to the survey also showed a high level of dedication to the game. Over 59% described Foldit as the only video game they played. Likewise, over 75% claimed to have been playing the game for over six months, while 49% claimed to play at least 15 hours a week. Furthermore, the top three reasons for continuing to play were to "make a contribution to science," to utilize a "background interest in science," and to engage in an "intellectual challenge." Interestingly, the actual game framework ranked near the bottom, with only three participants citing game play as a motivating factor [13].

H. SUMMARY

In this chapter, we detailed the history of volunteer computing, the power of distributed labor, and the impact on mass computing projects. Furthermore, we detailed two separate CSSG projects: Verigames and Foldit. We examined their approach to user involvement, player retention, and creation of scientifically relevant results that are seemingly opaque to the player. We detailed their successes, failures, and adaptations made to the games in order to improve gameplay, social aspects, and quality of outputted

work. Lastly, we included a primer on the most common metrics used to gauge the success of a game.

III. METHODOLOGY

A. INTRODUCTION

In this chapter, we will explain how we collected data from Foldit without direct access to the researcher's database. The first section will give an overview of how data metrics shaped the collection methodology. The second section will describe the layout of the Foldit site. The third section will show how a web scraper works and why we chose to create a custom scraping script. The fourth section will detail the placement and automation of the web scraper. The final section will examine the data storage methods we implemented.

B. DATA METRICS

Since Foldit is a free-to-play game, we decided to follow the same metrics that commercial free-to-play developers follow. For this thesis, we defined an active user as a user that scored at least one point. In order to score a point, a user must have attempted a puzzle. We implemented this definition for a few reasons. First, we wanted to capture the users who had an impact on the puzzle. For other CSSGs, minor contributions may have a measurable impact. For example, in Stormbound, a player contributes to formal software verification in code by detecting loop invariants [39]. A player casts "spells" to help uncover patterns in the game. In actuality, these spells act as new assertion statements that flag a piece of code for further processing. Since code can be broken into parts and tested individually, any new assertion is useful for the overall debugging effort. However, in Foldit granularity is not as fine. An incomplete puzzle equates to a molecularly unfavorable molecule. In biochemistry, an unfavorable molecule will almost never form a stable molecule or have any therapeutic value. Thus, a user creating an energetically impractical molecule would have the same impact as someone who never attempted a puzzle. However, the game would not allow an unfavorable to be submitted for evaluation. Thus, if a user created and submitted and candidate molecule, it can be inferred that their solution is plausible.

Second, citing privacy concerns, we were not allowed full access to user data. However, we were permitted to gather any publically available data. Unfortunately, public data lacked player session times and session counts. This information would have helped us better evaluate true daily activity. We would have also been able to evaluate the amount of time a player spends completing a puzzle, how many times they logged in, and other useful metrics that gauge player interest. In addition, work done by Tellioglu showed strong correlation in Verigames between user productivity and session time [16].

However, public data had significant information that allowed us to gauge player interest and contribution. As seen in Figure 3, each puzzle had its own page. Each page contained the puzzle's creation date, expiration date, difficulty level, and points for every player that submitted a puzzle solution. From this data, we could then extrapolate user contribution and specify when their activity occurred.

1. Whale Effect Graph

One hypothesis we wanted to test was if Foldit showed a strong reliance upon a small group of players in order to drive productivity. We followed the example previously used in Verigames [16], and plotted a Whale Effect Graph (WEG). This allowed us to visualize the contribution the top percentile of players had toward the overall productivity. As part of our collected data, we had scores for every puzzle that an active user attempted. In order to calculate the player contribution, we totaled the score for every puzzle that player attempted. We then divided that value by the total points scored from every puzzle. This gave us the following equation, where P_c is player contribution, P_p is player points, and T_p is total points:

$$P_c = \frac{\sum P_p}{\sum T_p}$$

Figure 3. Example of a Foldit Puzzle Scoreboard

Source [55]: *1157b: 60 Residue Monomer Design*. (2015, Nov. 19). Foldit. [Online]. Available: https://fold.it/portal/node/2001453

As can be seen in Figure 4, by plotting the productivity percentile on the y-axis and player percentile on the x-axis, we can then get a one-to-one representation of how every percentage of users has contributed to the overall score. For example, in Figure 1, we can state that the top 10% of users contributed around 85% of overall productivity.

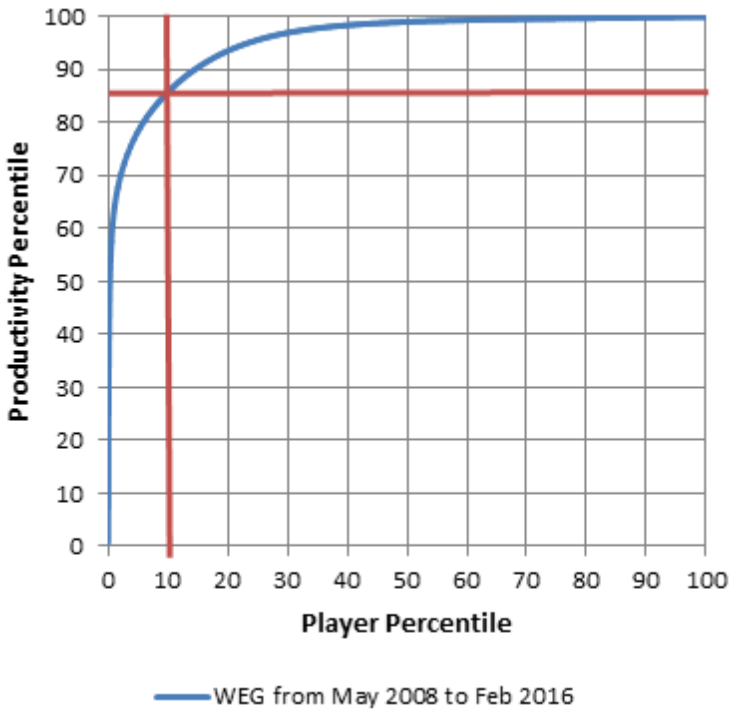

Figure 4. Whale Effect Graph (WEG)

2. Engagement Rate

Without a traditional daily active user (DAU) metric, we had to create a separate measurement to evaluate Foldit's "stickiness." In order to score a point, a user must have attempted a puzzle awarding points. We initially defined two groups from which we sampled our data: puzzle active users (PAU), defined as a user that attempted a given puzzle during a month, and the Monthly Active Users (MAU), the number of unique users that attempted puzzle during a given month. It should be noted that MAU is a set representation of PAU. While a user may attempt many puzzles in a month, they would only count once in the MAU. However, they would be counted for every puzzle attempt in the PAU.

In order to gauge "stickiness," we defined a new metric called task engagement rate (TER). If an individual played a game more than once, it was assumed that that player showed more interest than a player who only participated once. To calculate TER, we first separated puzzles into monthly blocks according to their expiration date. Next,

we found the MAU by combining each puzzle block and removing duplicate names. For each puzzle in the block, we then totaled the number of users. This gave us the PAU. We then divided each PAU by the MAU, thus giving a TER for each individual puzzle. Lastly, since we were interested in the overall TER for a given month, we took the average of the TERs. This gives us the following formula for monthly TER (mTER), where n represents the number of puzzles in a month:

$$mTER = \frac{\sum \frac{PAU}{MAU}}{n} * 100$$

While PAU and MAU alone represent the number of active users during their respective collection periods, TER gives us a defined rate and allows for trend analysis. A high TER is indicative of large amount of activity. If this high level is sustained over a significant amount of time, we can extrapolate that the game has been successful. Likewise, low or continually decreasing TER would indicate that players have lost interest in the game. At that point, the developers should think about either starting a new development cycle or implementing changes in order to try to recapture lost users.

C. FOLDIT WEBSITE

As stated earlier in the chapter, very early on in our study, we asked the Foldit developers for direct access to detailed player data. This database would have included all user demographics, participation levels, and historical scores. This would have allowed for an easier analysis from first hand sources. However, the developers citing privacy concerns denied this request. Since we were interested in only game scores for each individual user, we obtained permission to scrape their leaderboards and any other public facing information we might need.

The Foldit website was hosted by the Computer Science Department at the University of Washington. The site administrators used Drupal content management system to create and maintain the site [56]. The basics of Drupal relied upon the use of nodes, modules, added extensions, and templates. Every time content was created, that content was assigned a node. Node creation encapsulated every aspect of the site, from

posts in the discussion room to user account details. A module was akin to a programming library, adding functionality and creating additional non-native capabilities. Finally, a template acted as a style guide depending on the node type. The extensibility of Drupal allowed for easier site administration. However, the process of node creation added some difficulty to web scraping. An example of the Foldit site can be seen in Figure 5.

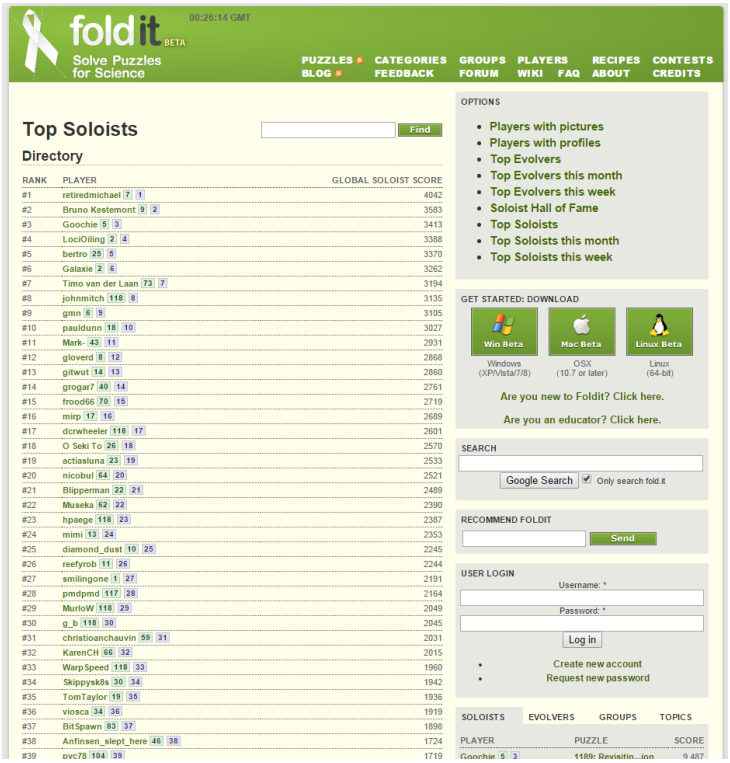

Figure 5. Foldit Soloist Leaderboard

Source [45]: *Soloist Hall of Fame*. (2016, Feb. 12). Foldit. [Online]. Available: http://fold.it/portal/players/s_all.

D. WEB SCRAPING

Web scraping is the process of gathering large amounts of data from a website, and storing it locally. Depending on the immensity of data, scraping is usually best accomplished using an automated process. A scraper imitates a regular web browser in that a website's markup language is parsed and interpreted. However, instead of displaying this information in a visual manner, the raw data is temporarily stored so

further filtering can occur. Once the raw page has been downloaded, one can customize a scraper to look for specific tags within the markup language. Tags themselves identify different elements of the site; they can demarcate tables, images, and have additional code such as JavaScript. Depending on the scraping library, one can emulate a wide variety of web browser capabilities including full JavaScript control, user-agent masking, and embedded element manipulation.

In order to investigate the layout of the site, we used the Firebug plug-in to look at the page source. The site used multiple embedded tables, making it more difficult to scrape the correct table. However, since the site used the same style sheet for each scoreboard page, a singular program could be used to iterate through each page with reliable results. Although several commercial programs were initially trialed, we ended up using a custom-made Python script for more refined results.

1. Scraping Code

The code was written with several distinct, mandatory capabilities in mind: the ability to read a web page, parse the entire website for the desired data and save that data to an external computer. A number of considerations were put into how to construct the web crawler. First, the code had to fully capture data from the website, and then discard unnecessary elements that did not contain user data. The code also needed to have proper exception handling. Given the volatile and ephemeral nature of websites, it was not possible to predict if the Foldit site would be down for maintenance, return an empty page, or have other unforeseen technical difficulties. Likewise, it was possible that a link might no longer exist, thus the scraper would become stuck in an endless loop and not iterate through the rest of the site. Finally, the scraper needed to know when to terminate itself.

As mentioned previously, the iterative node creation process used by Drupal added complications to the scraping code. Since node creation occurred in real-time, any new node was added sequentially. For example, a forum post and a puzzle page could be assigned consecutive nodes. Without proper grouping, we could not simply iterate from

page to page. Instead, we had to scrape the current page for the proper hyperlink to the next page.

Given the unfeasibility of completely reproducing the Foldit website, we acknowledged some trade-offs would need to be made. First, if a page failed to load, the program would retry the page after 60 seconds. If the page failed to fetch a second time, the failed link would be written to a log file for that day. The program would then attempt to load the subsequent page. One issue arising from continually attempting to a load a failed page was that the scraper might not realize that there are no further pages to scrape. During our initial testing, a quirk in the Foldit site was found that caused the scraper to be in an endless loop. If the scraper reached the final page on a scoreboard, the scraper would not terminate unless explicitly told to do so. We were expecting an exception to be raised if the scraper tried going beyond the final page. However, the Foldit site actually incremented the hyperlink, yet returned the final page. Thus, the scraper would never stop. To prevent this behavior, the first page was parsed for the hyperlink to the last page on the scoreboard. The scraper then saved this value as an integer. Whenever an attempt to scrape a page occurred, a counter would be incremented. This allowed for an accurate accounting of both failed and non-failed links. Once the counter value reached the stored final page value, the scraper would terminate.

2. BeautifulSoup

A scraping program can be written in any language capable of establishing sockets and connections. Given the ease of use, extensive module library, and multiplatform support, we chose to write our scraper in Python 2.7 using BeautifulSoup and the lxml library. BeautifulSoup is a Python module written specifically to help parse HTML and XML. The flexibility, easy implementation, and dense software documentation of both Python and BeautifulSoup provided for easier debugging and on-the-fly alterations. Our custom-made scraper also made automation much easier than an off-the-shelf solution. For example, we were only concerned with three elements on a total webpage: the rank, username, and score. Commercial scrapers would harvest the entire page and store it in a non-easily translatable manner, usually in UTF or UTF-8.

Instead, our code produced comma separated value (CSV) tables, a plain text representation of the data free of markup irregularities. This made our data much more robust and flexible.

E. DATA COLLECTION

Our data collection was divided into two periods. Our initial live data gathering began on June 1, 2015, and terminated on February 15, 2016. In the first phase of our study, we focused upon the daily and monthly soloist score for each user. It was our initial assumption that scores posted on these scoreboards were a raw accumulation of scores. We thought scores were calculated continuously and if a user improved enough on a given puzzle, their relative ranking would increase. As a result, a separate script was written for each leaderboard. Each individual script was then automated to be executed at 12 AM each day.

At the beginning of the study, over 12,000 pages of usernames and scores were present on the Foldit site. Given that each page had 50 users, this gave an initial approximation of 600,000 users who had registered an account since the beginning of the program. It was our intention to capture each user score for each day. However, given the large numbers of users and pages needing to be scraped each day, a full data capture was deemed unlikely. Thus, we deemed a 2% loss of data as being satisfactory for our results.

After latter analysis of our live data collection, it was apparent that our assumptions were incorrect. Thus, we conducted further site investigation so that we could better corroborate our data. A second source was found that had every created and completed puzzle starting from the inception of Foldit. Each puzzle page had creation date, expiration date, and difficulty level. In addition, the ranking, points, and total number of users display had a similar setup to the scoreboard seen in Figure 5. This made it easy to adapt our existing program to scrape additional portions of the website.

Given the tight firewall rules of the NPS network, it was decided to place the software outside of the NPS network in order to reduce the possibility of traffic filtering. The scraper was located on its own separate machine hosted from a Digital Ocean droplet, a cloud-based virtual computing company, running Ubuntu 14.10. This had the

additional theoretical benefit of having the scraping script closer to the host site since the droplet was positioned in a Seattle datacenter.

F. INFORMATION PARSING

With such a large number of CSVs, we had to create a method to organize the information. Each CSV only contained three pieces of information: the user rank, the user name, and the score. However, each leaderboard had every user that had ever registered an account, even if that user was inactive. In order to be as data complete as possible, we decided to try to collect data from every page on the leaderboard.

Initially, we tried to replicate the Drupal content management system used by Foldit. Unfortunately, using Drupal soon became unwieldy. Each user, representing a single data point, would be instantiated as a node. With such a large number of users, processing time soon exceeded the 24-hour period in which our script was running. Given the simplistic nature of the data, we decided to do a direct analysis using a custom python script.

Since the data collected consisted of nothing more than rank, name, and score, we created an alternative parsing solution. One of our main concerns was deriving the DAU, thus day-to-day analysis was put at the forefront. To facilitate this action, we wrote a CSV comparison tool that would detect differences in the number of users, changes in score for each user, and compute the cumulative score between two consecutive days. The final output would be written to a master CSV for easy importation into a spreadsheet program.

The comparison tool was coded in Python 2.7.10 and utilized the CSV module. The initial tool was written using two lists. However, it quickly became apparent that the sheer number of users would take a significant time. The tool was rewritten to use a Python dictionary to compare against a list. By casting one CSV into a dictionary, we were able to use a username as a key with the corresponding score from that day. The program would then compare a list generated from the second CSV, searching for matching keys from the dictionary.

When using the list method, iteration and list creation both took $O(n)$ time. Compounded, the time complexity ran in $O(n^2)$ time. Dictionaries transform the data into a hashmap. Since lookups in a dictionary take constant time, the loop comparing the CSVs is not dependent on rows. By implementing a dictionary, we were able to reduce time complexity to $O(n)$. This significantly reduced processing time.

If a match was found, a row would be added to a third CSV with the format: **[username, dayZ score, dayZ+1 score, (dayZ+1 score – dayZ score)]**. In this format, we could see if new users were added, there change in score, and if a user was active during that day. When combined with a spreadsheet program, we could then create a cumulative score from the date. Since each date had a score, we could then plot activity as cumulative score by active users per day.

Toward the end of our study, we decided to mine previously completed puzzles. While we were initially interested in Foldit data in the present term, it quickly became apparent that our previously scraped data gave insufficient insight into user performance. As was previously discussed, instead of focusing completely on leaderboard data, we discovered that every puzzle posted in Foldit was logged under the "Puzzles" category. In addition to the same type of player leaderboard, each puzzle had its own node and metadata tags. This proved useful since we were able to assign definitive dates to when a puzzle was deployed.

After some alterations to our scraping code (Appendix B), we were able to iterate through each page and record data for every puzzle starting from May 2008, the first instance of a score appearing, to the end of February 2016, the end of our study period. For each puzzle, the program created a separate CSV puzzle with user rank, name, global score, and total points. We then wrote two separate programs to group unique players. One program aggregated all players into a master list. Each time a player played a puzzle, we increased a counter that kept track of their puzzle attempts, and then increased their accumulated total score. This allowed us to evaluate the player contribution as a whole. This also gave us a definitive number of unique players that had played a puzzle. However, this accumulated data set provides no sense of how Foldit changed over time.

Thu, a second program sorted each puzzle into yearly periods. By evaluating yearly periods, we could then assess with more granularity. It should be noted that these two datasets would have differing numbers of players. In the total unique user data set, players were counted only once. In the second data set, a user was counted more than once if they appeared in separate yearly blocks. Thus, the total summation of the unique users from yearly periods will differ from the total number of unique users over the entirety of the sampling period.

G. DATA VERIFICATION

For live scraping, over 500,000 user scores were accumulated on a daily basis. With such a large dataset, manually verifying the correctness of each users score was unfeasible. As an alternative solution, we investigated a randomized subset of users to see if our scraped data matched the data available on the Foldit website. This proved useful for correcting errors in our scraping code and in enforcing confidence in our results.

A similar procedure was conducted on the scraped historical puzzle data. After collecting data on every completed puzzle, data was grouped into one-year periods. For each period, we randomly selected 20 users. We then manually verified the data by examining the user's profile for the relevant puzzle scores.

H. SUMMARY

In this chapter, we outlined the data collection method and data analysis techniques central to this thesis. We first described how data metrics shaped our approach to data collection. Next, we described the layout of the Foldit website. We further detailed how web scraping works and the specific alterations we made to scrape the Foldit site. Afterward, we defined how we conducted data collection, including web scraper automation and which portions of the Foldit site we scraped. Finally, we outlined how we parsed and stored information retrieved from the relevant Foldit pages.

IV. ANALYSIS AND EVALUATION

A. INTRODUCTION

This chapter will incorporate the relevant data collected during the duration of our study. We will first mention some limitations revealed in our data collection. Next, we will present data mined during our study. Afterward, we will present the relevant metrics as previously discussed in Chapter III. We will then compare these metric to previous work done on Verigames and other CSSGs. Finally, we will make some assertions about Foldit using our data prior work done on CSSGs.

B. LIMITATIONS

Since we were not granted access to the researcher's own data, it was questionable if our web scraping data accurately reflected ground truth. Live data collection over the web has inherent flaws that cannot always be accounted for. It was difficult to predict when the site would be unreachable. As such, we tried to account for as much error handling as possible. Our scraper code had exception handling and wait timers in case a page returned an error (See Appendix A).

In our scraper code, we anticipated possible pages being unavailable before scraping completed. As a result, retry timeouts were written such that if a page failed to load after a minute, a counter was incremented and the scraper was instructed to try the next page. This problem only affected the live data collection phase of our study. Since the historical puzzle data was static, we could retry as many times as possible in order to create a full snapshot of the data.

According to Foldit's user wiki page [57], the Foldit developers changed the scoring system such that the exponent used for calculating points decreased from seven to five. Points were derived from a fraction. Thus, decreasing the exponent had the effect of slightly increasing player points. This change might have been a result of feedback received from the player base and acted as a psychological boost in order to keep player interest.

Initially, we wondered if this change would have a significant impact on score. However, after close examination of Foldit's scoring system, it was revealed that score was more reliant upon a player's relative rank for a puzzle and the scoring change produced only minor changes.

Our data also lacked information useful for determining player interest. Session counts and session times would have provided additional insight into player behavior. Although we were able to account for the number of puzzles a player attempted, knowing how long a player spent playing a puzzle would be one indicator of prolonged interest and difficulty level. Likewise, we had no way of knowing how often a player logged in to play a particular puzzle. Instead, with only a gross number of attempted puzzles, we were limited in our evaluation.

C. USER DATA

An important aspect to a game's success is the ability for the game to attract and maintain a stable player base. As was described in Chapter II, traditional games usually follow a common game life cycle. Interest is strongest during the initial launch of the game followed by a sharp decrease in interest. Depending on the game genre, the developers may try to attempt to prolong productivity by releasing content updates. Alternatively, they may cease development and continue with development of other product lines.

Work done by Tellioglu [16] showed that Verigames had a similar game life cycle, albeit in an abbreviated, six-month time span. Both Verigames A and B had an 85% drop off in active users within a month of launch. Four months after launch, Verigame A only had 6% of the initial active user base, while Verigame B had effectively no active users.

Table 1 represents the total summation of active Foldit users from May 2008 to February 2016. To be counted amongst this group, a user must have attempted at least one scoring puzzle during this date range. Over the history of Foldit, 2011 scoring puzzles have been attempted. At the end of data collection, we counted 627781 total

registered accounts. A total of 123974 users have attempted a scoring puzzle, with their average score being 69.98 out of 100. On average, there were 284.98 users per puzzle.

Table 1. Cumulative FoldIt User Data—May 2008 through February 2016

Total Active Users	Total Registered Accounts	Total Puzzle Count	Avg Attempts	Avg Score (out of 100)	Avg Users Per Puzzle
123974	627781	2011	4.77	69.98	284.98

By dividing the number of users that attempted a scoring puzzle by the total number of registered accounts, we can see that 19.74% of all accounts attempted a puzzle. Although it may seem that only 19.74% of players played the game, our data lacks information concerning attempts at tutorial puzzles. These puzzles constitute 30 offline, unscored puzzles meant to train the user. If this data was included, we could then better evaluate user retention. Likewise, we could evaluate if a specific point in the game caused the user to quit.

Figure 6 represents a cumulative distribution function (CDF) of puzzle attempts conducted by Foldit users from May 2008 to February 2016. Combined with the data from Table 1, it is quite apparent that many users only play Foldit a few times before abandoning the game. Likewise, the number of users that play a disproportionate amount is quite small. It should be noted that while this CDF correctly demonstrates the activity of distinct percentages of users, it lacks any perspective concerning their contribution.

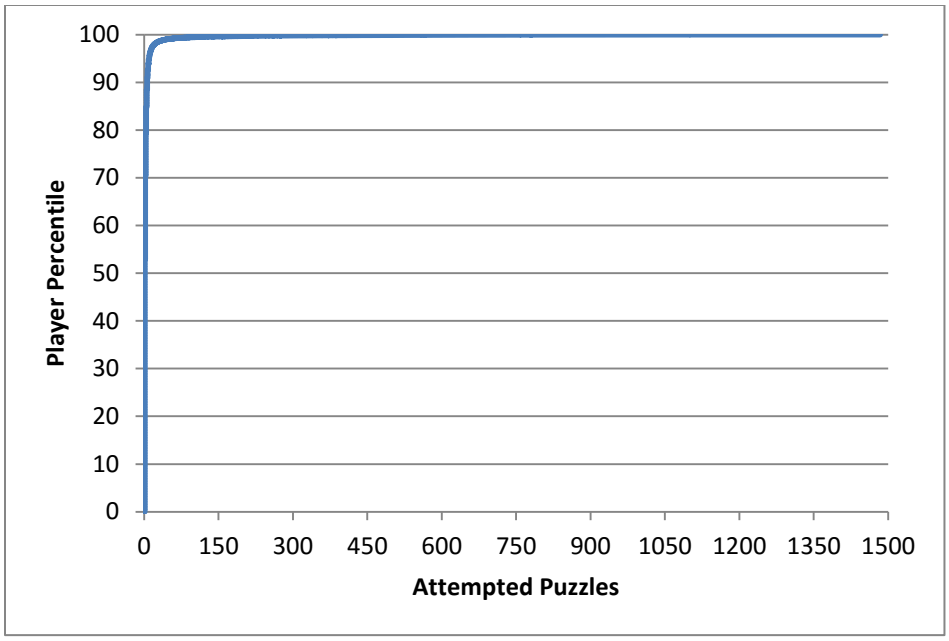

Figure 6. CDF of Puzzle Attempts—May 2008-February 2016

In Table 2, we tabulated puzzle data according to yearly blocks. It should be noted that the summation of the active users by year is different from the cumulative data. This is due to a given user being active in different years. The overall active user distribution for Foldit had an uncharacteristic cycle. From year 1 to years 2, Foldit saw the characteristic drop off in player activity, losing well over 60% of the active user base. However, from year 2 to year 3 and year 3 to year 4, the active user base expanded until reaching a max active player base of 40,606 unique users.

Although we cannot say with certainty what precipitated such a large increase in active users, a few notable events occurred during this time. For one, several major papers were published concerning Foldit. These papers established the benefits of crowd-sourcing protein design and included the previously mentioned modeling of an important HIV protein [47, 48, 58]. Around this time, news surrounding the success of Foldit trickled into the media. Reports by NBC [59], NPR [60], BBC [61], and other media outlets most likely helped drive interest in the game, thus boosting the total number of active users significantly. In addition, Foldit won several awards for game design, also likely helping drive interest in the game [62, 63].

After reaching its apex number of active users, Foldit saw a precipitous drop off in active users. However, during that period, the average number of puzzles a particular user attempted increased. One possible explanation for this result is that less interested players quit the game. This had the effect of concentrating user contribution into a smaller of players who were more interested and invested more time into the game.

Table 2. Foldit User Data by Year

Collection Period	Active Users	Total # of Puzzles	Min Attempts	Max Attempts	Avg # of Attempts	StDev
May '08 - '09	20058	314	1	217	4.64	11.63
May '09 - '10	7668	319	1	166	5.01	15.11
May '10 - '11	19300	218	1	151	3.31	8.78
May '11 - '12	40606	257	1	168	3.01	9.34
May '12 - '13	20092	245	1	190	3.89	14.11
May '13 - '14	12953	250	1	211	5.53	19.23
May '14 - '15	10223	251	1	210	6.06	21.5
May '15 - Feb '16	8332	157	1	140	5.06	15.06

Studies conducted by Curtis [13], Cooper [40], and Tellioglu [16] all showed a strong correlation between gaming session time and the productivity of the player. However, game industry analytics have constantly pointed to boredom and game difficulty as main drivers for losing players [64]. One overlooked aspect to time spent playing a game is the social interaction needed to drive player interest. While Verigames lacked social interaction, Foldit had an active wiki page, forums, and direct messaging. Players could also form teams, potentially providing encouragement and moral support when a particularly difficult puzzle was posted.

D. CURRENT USER PARTICIPATION LEVELS

The initial premise of this thesis relied upon determining DAU by collecting scoreboard data every day. It was assumed that the scoreboard was dynamic, meaning any scoring changes would be seen immediately. However, after further investigation, it was discovered that the leaderboard was updated only when a puzzle work period expired. Although a majority of puzzles opened and closed within a week, this was not always the case. As a result, the DAU data we gathered was inconsistent and did not provide the type of data we had originally hoped to analyze.

This data, however, had other uses that proved useful to our analysis. For one, historical puzzle data only recorded users who had attempted a puzzle. Using this data alone, there was no way to evaluate the number of users that created an account during a given period and did not attempt a puzzle. Likewise, without any timing data, there was no way to tell the experience level of a user attempting a puzzle. Furthermore, current data would be most useful since we were interested in how Foldit was performing in its current state.

Starting June 1 2015, 33291 new users created accounts. As seen in Figure 6, the number of newly registered users in relation to the total user count remained relatively constant. The largest portion of users was created during July and August 2015. From those 33291 new user accounts, 6562 new users attempted a scoring puzzle. This meant that 19.71% of newly registered accounts ended up attempting a puzzle. This nearly matched the average seen across the entirety of the Foldit project. We also accounted for 8332 total active users. This would indicate that 78% of players during this time were new players.

These statistics indicate that new players compose a significant portion of the current user base. It should be imperative to try to convert new players into returning players. However, with unusually similar user churn, Foldit seems to have reached a situation where the number of new players converting to returning players equates to the number of returning players leaving the game.

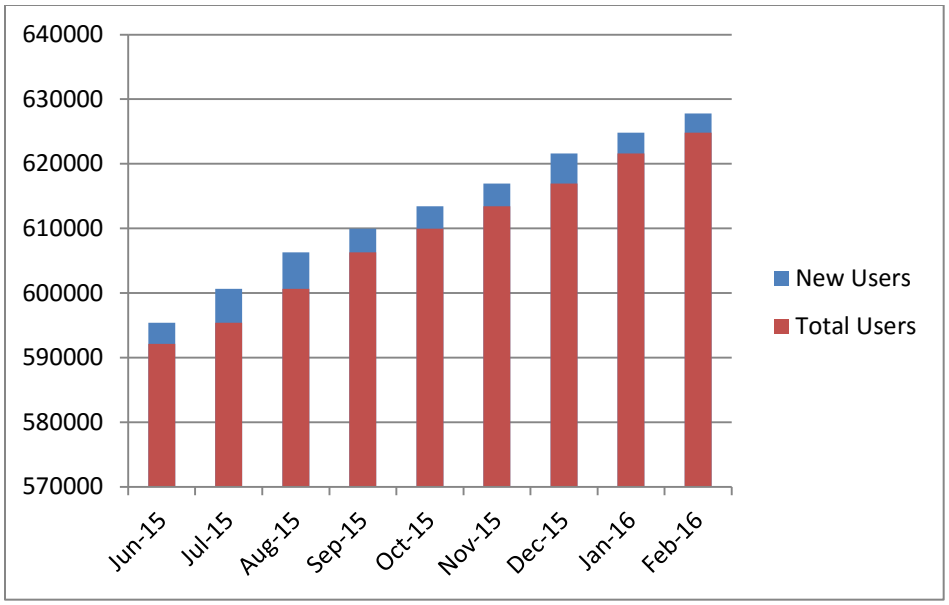

Figure 7. New Users Added Shown as a Portion of Total Users

Data in Table 3 demonstrates the difference in puzzle attempts by new and returning players. New players, on average, attempted significantly fewer puzzles than returning players. This is not surprising since returning players most likely return to a game because they derive some enjoyment from playing. Such a low number of average attempts also point at the difficulty in converting new players into consistent players. It should be noted that the sampling period could have possibly played a role in altering the data. For example, it is quite possible that several returning players played frequently during the May timeframe, but only once afterwards.

Returning players also scored significantly better and attempted more puzzles than new players. Unsurprisingly, this would indicate that experienced players are better at the game. Indeed, the fact that returning players, on average, attempted significantly more puzzles would seem to signify ongoing interest.

Table 3. New and Returning User Data—June 2015-February 2016

User Type	Total Users	Avg Score	Total Puzzles	Avg Attempted Puzzles	Min Attempts	Max Attempts	StdDev Attempts
New	6406	15.30	134	1.60	1	92	4.23
Returning	1770	311.29	134	14.82	1	140	29.68

Figure 8 shows the breakdown of players by percentile and their equivalent number of puzzle attempts. Much like the CDF shown in Figure 6, the CDF of current users shows a large proportion of users attempted fewer than 10 total puzzles. Likewise, this would indicate a small number of players show large exuberance and play the game at a very high frequency.

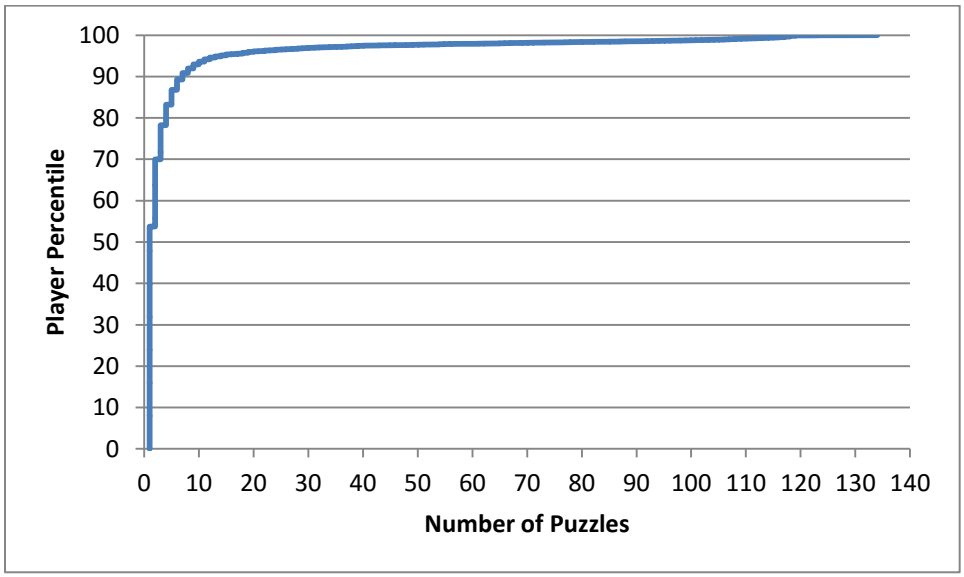

Figure 8. CDF of Puzzle Attempts—June 2015–February 2016

Figure 9 demonstrates the performance difference between new and returning users when charted on a WEG. Although both new and returning player bases demonstrate a strong disposition toward whales, the top 10 percent of returning players perform significantly better than the corresponding group of new players. Interestingly, as the player percentile increases, both new and returning player productivity converges quickly. However, it should be noted that productivity on a WEG is relative to the dataset being analyzed.

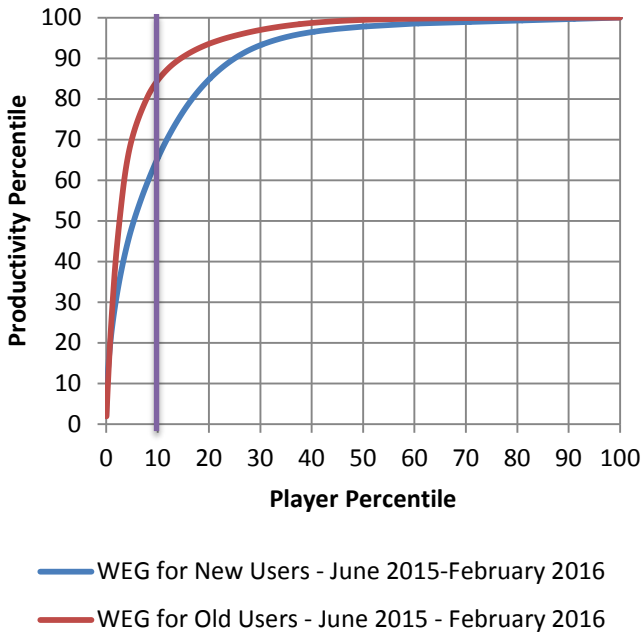

Figure 9. WEG of New Users and Old Users

E. TASK ENGAGEMENT RATE

An important part of any game is how well the game can retain players. A game's ER can reveal how often a player comes back to playing a game. As was described in Chapter II, players showed the most interest in a game during immediately after launch. At some point, the number of players plateaus, leading to an eventual decline. However, the period for each of these phases is highly variable. A bad game might see an abbreviated game cycle, with interest peaking within weeks of launch and a sharp decline of players. In contrast, a highly successful game may see a prolonged period of growth that can last months.

Traditionally, ER uses an average of DAU divided by MAU. The resulting metric returns how likely a player plays on a given day of a month. Since our data lacked DAU, we extrapolated player retention according to how often a player attempted a puzzle in a given month.

As was described in Chapter III, in an attempt to create a metric similar to traditional ER, we created TER, or task engagement rate. TER used PAU divided by

MAU to create a TER for each puzzle. A monthly TER was then calculated by taking the average of every puzzle TER. Although lacking the fine granularity of a DAU, PAU still provides a measurement of unique user activity per puzzle. In this regard, a higher TER would indicate a higher likelihood a given user will return to the game and give some measurement of the game's "stickiness." For a traditional game, an ER between 10 and 30 percent is indicative of high player interest regardless of raw player count. In comparison, Foldit's TER falls within these bounds of a successful game. For detailed data, please see Appendix E.

Table 4. TER of Foldit by Month

Month	# of Puzzles	Average PAU	MAU	Average Monthly TER
June	14	271.36	1146	23.7%
July	18	279.28	1615	17.3%
August	16	275.63	1597	17.3%
September	17	228.76	1110	20.6%
October	16	333.44	2186	15.3%
November	13	300.23	1599	18.8%
December	18	281.17	1672	16.8%
January	16	223.25	932	24.0%
February	15	243.20	1277	19.0%
Averages	15.89	270.70	1459.33	19.2%

Another measurement of user retention was to count how often a player played multiple puzzles. Although the number of puzzles in a month varied, during our sampling period, each month had at least 14 puzzles. We delineated each group into three categories: users who attempted 1–2 puzzles, users who attempted 3–6 puzzles, and users who attempted 7 or more puzzles. We reasoned that a player who only attempted a few puzzles had the lowest interest, while players who attempted many puzzles must have shown significant interest in the game. The remaining group, then, had average interest. Afterward, we counted the number of users per month that fell within each bucket.

Table 5 shows the outcomes from this tabulation. The vast majority of players showed low to medium amounts of interest, while a minority of players showed great interest.

Table 5. Foldit Puzzle Attempts by AU per Month

Month	# of Puzzles	1-2	3-6	7+	1-2	3-6	7+
June	14	738	218	189	64.5%	19.0%	16.5%
July	18	1123	264	228	69.5%	16.4%	14.1%
August	16	1175	224	198	73.6%	14.0%	12.4%
September	17	685	235	190	62.0%	21.0%	17.0%
October	16	1630	352	204	74.6%	16.1%	9.3%
November	13	1205	208	186	75.4%	13.0%	11.6%
December	18	1135	289	218	69.1%	17.6%	13.3%
January	16	559	172	201	60.0%	18.5%	21.5%
February	15	916	173	188	71.7%	13.5%	14.8%
Averages	15.89	1018.44	237.22	200.22	68.9%	16.6%	14.5%

Using these results alone, Foldit shows a strong, active user base in comparison to data shown for Verigames [16]. While Verigames showed an ER less than 5%, Foldit showed a sustainable amount of churn for the time being. In addition, data from Table 2 showed a continuous decrease in active users per year since its highpoint in 2011. However, the large number of players attempting more than seven puzzles is indicative of meaningful player interest and loyalty.

F. WHALE EFFECT GRAPH RESULTS

Previous work, as shown in Figure 10, by Tellioglu [16] showed Verigames A had a strong whale effect, while Verigames B had a much weaker whale effect. In Verigame A, the top 10% of players produced slightly less than 70% of productivity. In Verigame B, the effect was even weaker with the top 10% producing less than 45% of productivity. As shown in Figure 11, work done by Sauermann and Franzoni saw a disproportionately large contribution performed by a relatively small segment of the player base [65]. In their study, seven different CSSGs used the same platform, but had

different principal investigators and scientific purposes. In these games, players were presented with visual astronomical data, such as deep field telescopic images or high definition lunar photographs. A user was then tasked with distinguishing unique features as described by the rules of the game. Sauermann and Franzoni took this data and plotted it on a Lorenz graph, an economic graph that portrays the percentage of cumulative users and their corresponding contribution. This type of graph has similar features to a WEG, but plots the lowest contributors first. In contrast, a WEG emphasizes the top contributors and the data set is plotted accordingly. As such, in comparison the top 10% of a WEG is equivalent to the lower 90% of a Lorenz graph.

After plotting the data, the top 10% of players across all projects contributed between 71% and 88%. When compared against this data set, Foldit data showed the second highest WEG. As can be seen in Figure 12, over the entire history of Foldit, the top 10% of players produced nearly 85% of productivity. Puzzle data correlating to our 9 month live data collection also showed a strong WEG on par with the WEG calculated across the entire project.

Figure 13 charts the changing player productivity per year. When examining the graph, it becomes apparent that Foldit has been heavily reliant upon whales since the inception of the game. In comparison to Verigames, the top 10 percent of Foldit players produced at least 76% of all productivity. Although this may seem like an over reliance upon a few players, the lengthy learning curve show a game design built to turn its non-expert player base into a highly productive workforce.

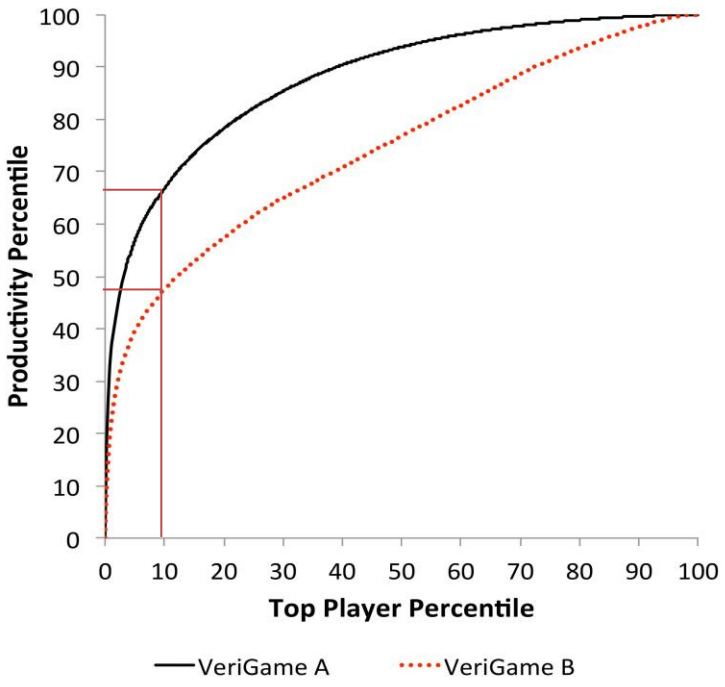

Figure 10. Verigames WEG

Adapted from [16]: U. Tellioglu. Quantifying the effectiveness of crowd-sourced serious games. M.S. thesis, Dept. Info. Sci., Naval Postgraduate School, Monterey, CA, 2014.

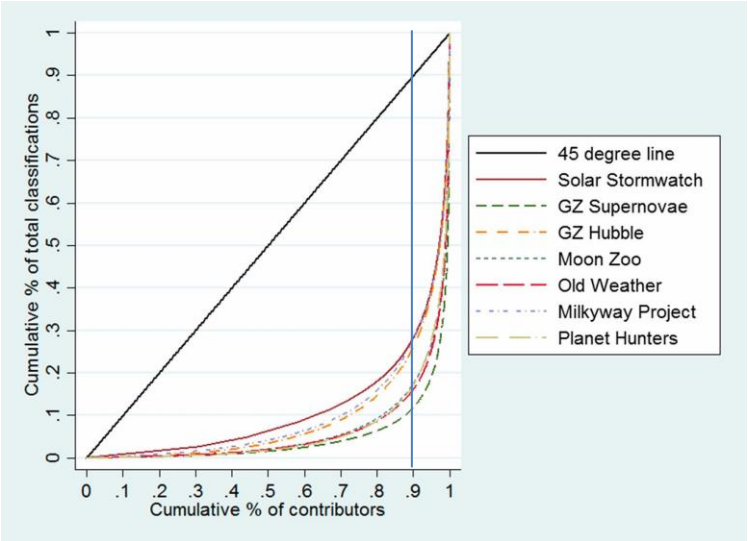

Figure 11. Lorenz Graph of Other CSSGs

Adapted from [65]: H. Sauermann and C. Franzoni, "Crowd science user contribution patterns and their implications," Proc. Natl. Acad. Sci. U. S. A., vol. 112, pp. 679–684, Jan 20, 2015.

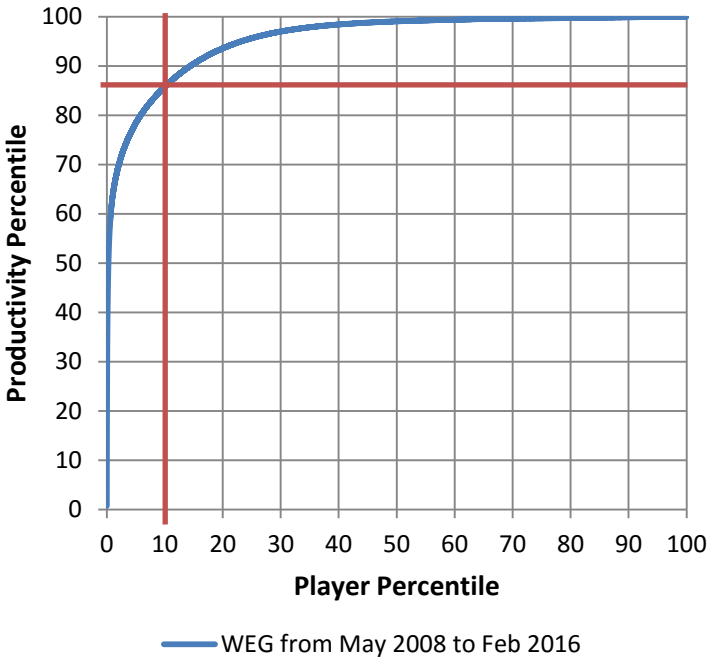

Figure 12. WEG from Cumulative Foldit Data

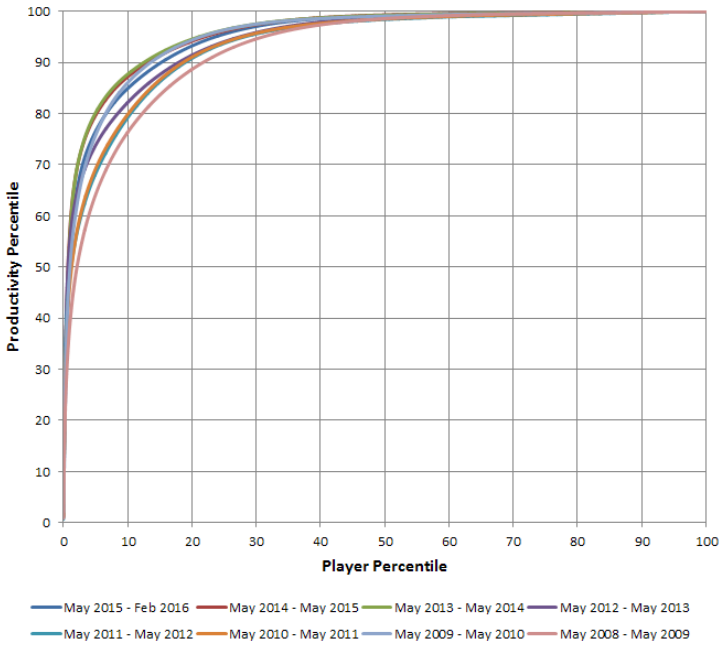

Figure 13. Foldit WEG divided into yearly time periods

G. SOCIAL INTERACTIONS, GAME DESIGN, AND PLAYER RETENTION

Since the inception of Foldit, two published player surveys revealed how player motivation can change over time. In Cooper's original thesis dissertation [40], 75% of player motivation originated in scientific purpose and game immersion. This is to say that most respondents initially played the game because of interest in protein folding, supporting citizen science, trying to solve game puzzles, and entertainment. The remaining 25% was split between achievement and social interaction, with achievement accounting for over 15% of the remaining group. This would indicate that social interactions were an ancillary consideration for player motivation.

A follow up survey conducted by Curtis [13] in the summer of 2012 revealed that 60% of players were motivated by making a contribution to science, while background interest in science and intellectual curiosity also ranked high. A third of players noted the social aspects, such as the online community and player-to-player interaction, as motivating factor. Interestingly, fewer than 10% of respondents ranked game play, fun, and solving puzzles as a significant motivating factor. Furthermore, over 60% felt that no other incentives, such as rewards or achievements, should be offered.

From this information, it becomes apparent that social interaction becomes more important as a game matures. In order to keep mature players interested and contributing to a project, peer reinforcement appears to form a strong motivating factor. Indeed, from our data set, the top performing users all belonged to a team.

From our data, we have shown the benefits of creating an experienced user base. While a preponderance of whales may seem like a detriment, for an older game like Foldit, invested users contribute vastly more than when the game is newer. In comparison to traditional games, the whale effect may be even more exaggerated in CSSGs. For developers of traditional games, their goal is to provide the highest entertainment at the highest profit margin. In contrast, a developer of a CSSG should aim to develop the most productive user base as possible, even if that user base is relatively small.

H. SUMMARY

In this chapter, we have described the limitations of our data collection, provided data analysis using our chosen metrics, and hypothesized about reasons why Foldit continues to persist as a CSSG. Our data revealed a large number of users creating accounts, with approximately 20% being converted into contributing users. The majority of puzzle contributions relied upon experienced, highly motivated players. This percentage was greater than those found in traditional games and other CSSGs.

V. CONCLUSION AND FUTURE WORK

A. SUMMARY OF WORK

With the top 10% contributing over 85% of total contribution, Foldit shows a strong whale effect. This would indicate that Foldit continues to retain a significant numbers of experienced players. Likewise, the TER over the past nine months is relatively high, with an average of 30% of players attempting more than three puzzles a month. Although an approximation of ER, a high TER provides evidence Foldit still attracts players that continue to play the game.

Foldit's use of team collaboration, social interactions, successful publicity, and lengthy tutorials all contribute to molding a highly capable core player base of 400 active users. Data shows this player base attempts the most puzzles a month and contributes more than 80% of total productivity. While this player base is strong in solving Foldit puzzles, it is unlikely the same players can successfully contribute to other CSSGs in a similar manner. Instead, these players are more likely to be successful at games similar in style to Foldit, such as games with 3D modeling and immediate visual feedback.

In comparison to how current CSSGs are designed, future CSSGs should try to emulate the Foldit model. While previous studies correctly surmised that low ERs are a significant obstacle for wider adoption of CSSGs, the poor results from those CSSGs are most likely caused by poor game design, weak marketing, or targeting the wrong type of player. In particular, the various Verigames projects may have aimed to capture the casual game market by hiding too much of the hard science. However, these games may have been better off trying to reveal more about how each player is contributing to the overall project rather than focus on a fictional narrative or flashy graphics. Thus, instead of attracting and retaining the type of player attracted to CSSGs, Verigames may have actually caused disinterest amongst those most interested in this type of game.

B. FUTURE RESEARCH

Foldit had the benefit of a plethora of publically available data. Still, stronger assertions can be made with finer grained data. For instance, if we knew how long a

player spent on a puzzle or the number of times a player opened a puzzle, we would be able to create an ER without having to resort to a coarser grained metric like TER. We also did not study players who registered an account, but never attempted a scoring puzzle. If we were to mine data concerning players who only played tutorial puzzles, we would be able to evaluate when a subset of the player base began to lose interest.

While this thesis emphasized individual play, Foldit also has team scores. Analyzing team data and individual contributions toward total teamwork may reveal novel approaches to CSSG design. Since social interactions and teamwork seemed integral for keeping players and improving individual contribution, Foldit in many ways resembles traditional team-based role-playing games, such as World of Warcraft. Stronger assertions about CSSG game design can be made if comparisons were made against team-based games, rather than non-similar CSSGs.

APPENDIX A. PYTHON CODE FOR SOLOIST SCRAPER

```
#!/usr/bin/python
import os
import requests
import csv
import time
import sys
from bs4 import BeautifulSoup
from requests.exceptions import ConnectionError

def get_num(x):
   return int(‘‘.join(ele for ele in x if ele.isdigit()))

next_num = 0
next_page = 0
page = raw_input(“Enter page # or ‘Enter’ for beginning”)

if(page == ‘‘):
   url_next = ‘http://fold.it/portal/players/s_all’
   counter = 0
else:
   url_next = ‘http://fold.it/portal/players/s_all?page=‘ + page
   counter = int(page)

url_last = ‘‘
print url_next

today_string = time.strftime(‘%m_%d_%Y’)
location = ‘/home/jay/Dropbox/Thesis/data/’ + ‘daily_soloist_’ + today_string + ‘.csv’
log = ‘/home/jay/Dropbox/Thesis/log/’ + ‘log_daily_soloist_’ + today_string + ‘.txt’

with open(location, ‘a’) as my_csv:
  while True:
   try:
      soup = BeautifulSoup(requests.get(url_next).text, “lxml”)
   except ConnectionError:
      time.sleep(60)
      soup = BeautifulSoup(requests.get(url_next).text, “lxml”)

   counter += 1

   if(url_last==‘‘):
      last_link = soup.find(class_=‘active’, title = ‘Go to last page’)
```

```
    url_last = "http://fold.it" + last_link['href']
    last_page = get_num(url_last)

print counter, last_page
log_write = open(log, 'a')
log_write.write(str(counter) + " " + str(last_page) + "\r\n")


for row in soup('tr', {'class':'even'}):
  cells = row('td')

  #current rank
  rank = cells[0].text
  import pdb; pdb.set_trace()
  #finds first text node - user name
  name = cells[1].a.find(text=True).strip()

  #separates ranking
  #rank1, rank2 = cells[1].find_all("span")

  #total global score
  score = row('td')[2].string

  data = [[int(str(rank[1:])), name.encode('ascii', 'ignore'), int(str(score))]]

  #writes to csv
  database = csv.writer(my_csv, delimiter=',')
  database.writerows(data)
print url_next + " completed scraping"
log_write = open(log, 'a')
log_write.write(url_next + "\r\n")

if(counter == last_page):
  print "Scraping Complete"
  sys.exit()

next_link = soup.find(class_='active', title='Go to next page')
while(next_link is None):
    print "Failed link. Trying to grab from " + url_next
    next_num = get_num(url_next) + 1
    log_write = open(log, 'a')
    log_write.write("Failed link. Trying to grab from " + url_next + "\r\n")
    url_next = "http://fold.it/portal/players/s_all?page=" + str(next_num)
    try:
        soup = BeautifulSoup(requests.get(url_next).text, "lxml")
```

```
    except ConnectionError:
        time.sleep(60)
        soup = BeautifulSoup(requests.get(url_next).text, “lxml”)
    next_link = soup.find(class_=‘active’, title=‘Go to next page’)

    if (counter > last_page):
        print “error fetching pages”
        sys.exit()

    counter += 1

if(next_num == 0):
    url_next = “http://fold.it” + next_link[‘href’]
    print url_next

log_write = open(log, ‘a’)
log_write.write(url_next + “\r\n”)

if(next_page == last_page):
  print “Scraping Complete”
  sys.exit()

next_num = 0
next_page = get_num(url_next)
```

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX B. PYTHON CODE FOR PUZZLE SCRAPER

```
#!/usr/bin/python
import locale
import requests
import csv
import sys
from bs4 import BeautifulSoup
from requests.exceptions import ConnectionError

def get_num(x):
   return int(‘‘.join(ele for ele in x if ele.isdigit()))

locale.setlocale(locale.LC_ALL, ‘en_US.UTF-8’)
next_num = 0
next_page = 0
pageB = raw_input(“Enter Start Page: “)
pageE = raw_input(“Enter End Page: “)

log = ‘/root/Dropbox/Thesis/puzzle_log.txt’

if pageB > pageE:
        print “Start page greater than end page. Please try again.”
        sys.exit()

for page in range(int(pageB), int(pageE)):
   print page
   f1 = open(log,’a’)
   #User entry for page number from “Puzzles” section
   url = ‘http://fold.it/portal/puzzles?page=‘ + str(page)
   puzzleSoup = BeautifulSoup(requests.get(url).text, “lxml”)
   puzzles = []

   #creates a list of nodes for each puzzle on page
   for row in puzzleSoup(‘div’, {‘class’:’name’}):
      link = row.a[‘href’]
      nodeNum = link.rsplit(‘/’,1)[1]
      puzzleNum = row.find(‘a’).find(text=True)[:4]
      puzzles.append((nodeNum, str(puzzleNum)))

   #loops through each node on a page corresponding to a puzzle
   for i in range(len(puzzles)):

   #node number
```

```
node = puzzles[i][0]

#puzzle number
puzName = puzzles[i][1]
url_next = 'http://fold.it/portal/node/' + node + '/show_players/'
url_last = ''

#creates csv
location = '/root/Dropbox/Thesis/puzzles2/'+ puzName + '-' + node + '.csv'

with open(location, 'a') as my_csv:
   while True:
      try:
         soup = BeautifulSoup(requests.get(url_next).text, "lxml")
      except ConnectionError:
         time.sleep(60)
         soup = BeautifulSoup(requests.get(url_next).text, "lxml")

      if(url_last==''):
         last_link = soup.find(class_='active', title = 'Go to last page')
         try:
            url_last = "http://fold.it" + last_link['href']
            last_page = get_num(url_last)
         except:
            f1.write(url_next + '\n')
            break
      try:
         playerData = soup('div', {'class':'view-content view-content-adobe-puzzle-
players-embedded'})
      except:
         break
      for table in playerData:
         for row in table('tr', {'class':'odd'}):
            try:
               cells = row('td')
               rank = cells[0].string
               name = cells[1].find_all('a')[1].find(text=True).strip()
               score = row('td')[3].string
               points = row('td')[4].string
               data = [[int(str(rank)), name.encode('ascii', 'ignore'), \
locale.atoi(str(score)), int(str(points))]]
            except:
               f1.write(url_next + '\n')
               break
            #writes to csv
```

```
            database = csv.writer(my_csv, delimiter=',')
            database.writerows(data)

        for row in table('tr', {'class':'even'}):
            try:
                cells = row('td')
                rank = cells[0].string
                name = cells[1].find_all('a')[1].find(text=True).strip()
                score = row('td')[3].string
                points = row('td')[4].string
                data = [[int(str(rank)), name.encode('ascii', 'ignore'), \
locale.atoi(str(score)), int(str(points))]]
            except:
                f1.write(url_next + '\n')
                break
            #writes to csv
            database = csv.writer(my_csv, delimiter=',')
            database.writerows(data)

    if(next_page == last_page):
        print "Scraping Complete"
        break

    next_link = soup.find(class_='active', title='Go to next page')

    if(next_num == 0):
        url_next = "http://fold.it" + next_link['href']

    next_num = 0
    next_page = get_num(url_next)
```

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX C. PYTHON CODE FOR CSV COMPARISON TOOL

```
#!/usr/bin/python
import csv

x = raw_input("Enter date for file 1: ")
y = raw_input("Enter date for file 2: ")

csvfile = open(myfile, 'r')
reader = csv.reader(csvfile, delimiter = ',')
#my_dict = {rows[1]:rows[2] for rows in reader}
my_dict = {}
for row in reader:
        if row[1] in my_dict:
                continue
        else:
                my_dict[row[1]] = row[2]

csvfile2 = open(myfile2, 'r')
reader2 = csv.reader(csvfile2, delimiter = ',')
my_list2 = list(reader2)

with open(myfile2, 'rb') as csvfile2:
  with open(location, 'a') as my_csv:
    reader2 = csv.reader(csvfile2, delimiter = ',')

    for row in my_list2:
      index = my_dict.get(row[1])
      if index is not None:
        if isinstance(int(my_dict[row[1]]),int):
          value1 = int(my_dict[row[1]])
        if isinstance(int(row[2]), int):
          value2 = int(row[2])
          value3 = value2 - value1
        user = [row[1],value1, value2, value3]
        data = csv.writer(my_csv, delimiter = ,"")
        data.writerow(user)


csvfile.close()
csvfile2.close()
my_csv.close()
```

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX D. PYTHON CODE FOR CSV COMBINING TOOL

```
#!/usr/bin/python
import csv
import os

csvList = []
nameDict = {}
rDir = "/home/jay/Desktop/Puzzles"
location = '/home/jay/Desktop/Puzzles/PuzzlesCombined.csv'

#traverses directory and gets all file names
for dirName, subdirList, fileList in os.walk(rDir):
    for fname in sorted(fileList):
        csvList.append(fname)

file1 = rDir + "/" + csvList[0]
csvfile = open(file1, 'r')
reader = csv.reader(csvfile, delimiter = ',')
for rows in reader:
    nameDict = {rows[1]: [int(rows[3]),1] for rows in reader}

#iterates from one csv to the next
for i in range(1,len(csvList)):
    file1 = rDir + "/" + csvList[i]
    csvfile = open(file1, 'r')
    reader = csv.reader(csvfile, delimiter = ',')

    mylist = list(reader)
    for row in mylist:
        if row[1] in nameDict:
            score = int(row[3]) + nameDict[row[1]][0]
            count = nameDict[row[1]][1] + 1
            nameDict[row[1]] = [score, count]
        else:
            nameDict[row[1]] = [int(row[3]),1]

my_csv = open(location,'a')
for key, value in sorted(nameDict.items()):
    my_csv.write(str(key) + "\t" + str(value[0]) + "\t" + str(value[1]) + '\n')
```

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX E. PUZZLE DATA PER MONTH—JUNE 2015-FEB 2016

Table 6. June 2015 Puzzle Data

Puzzle name	Users	Avg TER
1094-2000778.csv	279	0.243455
1095-2000785.csv	283	0.246946
1096-2000790.csv	224	0.195462
1097-2000800.csv	247	0.215532
1098-2000801.csv	232	0.202443
1099-2000813.csv	257	0.224258
1100-2000824.csv	259	0.226003
1101-2000838.csv	273	0.23822
1102-2000840.csv	236	0.205934
1103-2000842.csv	231	0.201571
1104-2000844.csv	284	0.247818
Begi-2000664.csv	165	0.143979
Begi-2000685.csv	286	0.249564
Begi-2000712.csv	543	0.473822

Total unique Users	Monthly TER	Min Puzzle Attempts	Max Puzzle Attempts	Avg Puzzle Attempts	Median Puzzle Attempts	StdDev
1146	.236786	165	543	271.3571	258	84.71455

Table 7. July 2015 Puzzle Data

Puzzle Name	Users	Avg TER
1105-2000856.csv	217	0.134365
1106-2000862.csv	242	0.149845
1107-2000886.csv	264	0.163467
1108-2000902.csv	269	0.166563
1109-2000906.csv	236	0.14613
1110-2000923.csv	263	0.162848
1111-2000926.csv	278	0.172136
1112-2000937.csv	303	0.187616
1113-2000945.csv	253	0.156656
1114-2000953.csv	251	0.155418
1115-2000961.csv	224	0.1387
1116-2000966.csv	251	0.155418
1117-2000967.csv	286	0.17709
Begi-2000745.csv	251	0.155418
Begi-2000771.csv	197	0.121981
Begi-2000795.csv	836	0.517647
Begi-2000819.csv	165	0.102167
Begi-2000841.csv	241	0.149226

Total unique Users	Monthly TER	Min Puzzle Attempts	Max Puzzle Attempts	Avg Puzzle Attempts	Median Puzzle Attempts	StdDev
1615	.172927	165	836	279.2778	251	142.6088

Table 8. August 2015 Puzzle Data

Puzzle Name	Users	AVG ER
1118-2000977.csv	212	0.132749
1119-2000983.csv	248	0.155291
1120-2000995.csv	256	0.160301
1121-2001015.csv	241	0.150908
1122-2001011.csv	213	0.133375
1123-2001022.csv	248	0.155291
1124-2001038.csv	240	0.150282
1125-2001044.csv	220	0.137758
1126-2001054.csv	229	0.143394
1127-2001066.csv	239	0.149656
1128-2001074.csv	223	0.139637
1129-2001081.csv	245	0.153413
Begi-2000859.csv	398	0.249217
Begi-2000903.csv	304	0.190357
Begi-2000929.csv	122	0.076393
Begi-2000955.csv	772	0.483406

Total unique Users	Monthly TER	Min Puzzle Attempts	Max Puzzle Attempts	Avg Puzzle Attempts	Median Puzzle Attempts	StdDev
1597	.172589	122	772	275.625	240.5	143.3959

Table 9. September 2015 Puzzle Data

Puzzle Name	Users	Avg TER
1130-2001087.csv	59	0.053153
1130-2001090.csv	206	0.185586
1131-2001101.csv	208	0.187387
1132-2001113.csv	217	0.195495
1133-2001117.csv	229	0.206306
1134-2001124.csv	217	0.195495
1135-2001145.csv	294	0.264865
1136-2001147.csv	283	0.254955
1137-2001175.csv	366	0.32973
1138-2001183.csv	225	0.202703
1139-2001185.csv	275	0.247748
1140-2001209.csv	227	0.204505
1141-2001215.csv	276	0.248649
Begi-2000974.csv	109	0.098198
Begi-2000992.csv	180	0.162162
Begi-2001020.csv	370	0.333333
Begi-2001046.csv	148	0.133333

Total unique Users	Monthly TER	Min Puzzle Attempts	Max Puzzle Attempts	Avg Puzzle Attempts	Median Puzzle Attempts	StdDev
1110	.206094	59	370	228.7647	225	80.48876

Table 10. October 2015 Puzzle Data

Puzzle	Users	Avg TER
1142-2001234.csv	292	0.133577
1143-2001249.csv	259	0.118481
1144-2001255.csv	287	0.13129
1145-2001287.csv	275	0.125801
1146-2001291.csv	247	0.112992
1147-2001302.csv	37	0.016926
1147-2001306.csv	258	0.118024
1148-2001308.csv	262	0.119854
1149-2001311.csv	276	0.126258
1150-2001331.csv	258	0.118024
1151-2001338.csv	276	0.126258
Begi-2001075.csv	168	0.076853
Begi-2001100.csv	1203	0.55032
Begi-2001130.csv	178	0.081427
Begi-2001164.csv	556	0.254346
Begi-2001191.csv	503	0.230101

Total unique Users	Monthly TER	Min Puzzle Attempts	Max Puzzle Attempts	Avg Puzzle Attempts	Median Puzzle Attempts	StdDev
2186	.152533	37	1203	333.4375	268.5	260.4117

Table 11. November 2015 Puzzle Data

Puzzle	Users	AVG ER
1152-2001344.csv	472	0.295184
1153-2001366.csv	261	0.163227
1154-2001404.csv	265	0.165729
1155-2001407.csv	229	0.143215
1156-2001433.csv	266	0.166354
1157-2001453.csv	224	0.140088
1158-2001472.csv	282	0.17636
1159-2001478.csv	249	0.155722
1160-2001489.csv	233	0.145716
Begi-2001220.csv	203	0.126954
Begi-2001253.csv	118	0.073796
Begi-2001294.csv	935	0.58474
Begi-2001318.csv	166	0.103815

Total unique Users	Monthly TER	Min Puzzle Attempts	Max Puzzle Attempts	Avg Puzzle Attempts	Median Puzzle Attempts	StdDev
1599	.187762	118	935	300.2308	249	207.2499

Table 12. December 2015 Puzzle Data

Puzzle Name	Users	AVG TER
1161-2001509.csv	267	0.159689
1162-2001512.csv	283	0.169258
1163-2001514.csv	242	0.144737
1164-2001540.csv	255	0.152512
1165-2001552.csv	261	0.1561
1166-2001566.csv	226	0.135167
1167-2001581.csv	267	0.159689
1168-2001603.csv	231	0.138158
1169-2001651.csv	214	0.12799
1170-2001664.csv	256	0.15311
1171-2001672.csv	216	0.129187
1172-2001674.csv	210	0.125598
1173-2001692.csv	220	0.131579
Begi-2001342.csv	252	0.150718
Begi-2001382.csv	483	0.288876
Begi-2001408.csv	182	0.108852
Begi-2001440.csv	153	0.091507
Begi-2001476.csv	843	0.504187

Total unique Users	Monthly TER	Min Puzzle Attempts	Max Puzzle Attempts	Avg Puzzle Attempts	Median Puzzle Attempts	StdDev
1672	.168162	153	843	281.1667	247	155.354

Table 13. January 2016 Puzzle Data

Puzzle Name	Users	Avg TER
1174-2001693.csv	267	0.286481
1175-2001694.csv	205	0.219957
1176-2001695.csv	219	0.234979
1177-2001697.csv	227	0.243562
1178-2001723.csv	245	0.262876
1179-2001734.csv	208	0.223176
1180-2001738.csv	233	0.25
1181-2001757.csv	243	0.26073
1182-2001767.csv	272	0.291845
1183-2001779.csv	229	0.245708
1184-2001783.csv	242	0.259657
1185-2001794.csv	250	0.26824
Begi-2001513.csv	133	0.142704
Begi-2001548.csv	187	0.200644
Begi-2001582.csv	285	0.305794
Begi-2001671.csv	127	0.136266

Total unique Users	Monthly TER	Min Puzzle Attempts	Max Puzzle Attempts	Avg Puzzle Attempts	Median Puzzle Attempts	StdDev
932	.239539	127	285	223.25	231	44.33434

Table 14. February 2016 Puzzle Data

Puzzle Name	Users	AVG TER
1186-2001803.csv	254	0.1989037
1187-2001822.csv	209	0.1636648
1188-2001825.csv	236	0.1848081
1189-2001845.csv	245	0.1918559
1190-2001857.csv	223	0.174628
1191-2001860.csv	191	0.1495693
1192-2002051.csv	229	0.1793265
1193-2002056.csv	208	0.1628818
1194-2002058.csv	251	0.1965544
1195-2002081.csv	225	0.1761942
1196-2002089.csv	220	0.1722788
Begi-2001696.csv	102	0.0798747
Begi-2001709.csv	706	0.5528583
Begi-2001729.csv	119	0.0931872
Begi-2001759.csv	230	0.1801096

Total unique Users	Monthly TER	Min Puzzle Attempts	Max Puzzle Attempts	Avg Puzzle Attempts	Median Puzzle Attempts	StdDev
1277	.1904464	102	706	243.2	225	135.37419

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF REFERENCES

[1] J. Burt. (2015, Nov 11). Michael Dell: PC growth strong in "Post-PC Era" *eWeek* [Online]. Available: http://www.eweek.com/pc-hardware/michael-dell-pc-growth-strong-in-post-pc-era.html

[2] D. Hutcheson. (2015. Apr 2). Transistor production has reached astronomical scales. *IEEE Spectrum* [Online]. Available: http://spectrum.ieee.org/computing/hardware/transistor-production-has-reached-astronomical-scales

[3] *Intel Core i7-6700K Processor* (2015, October 15). Intel. [Online]. Available: http://ark.intel.com/products/88195/Intel-Core-i7-6700K-Processor-8M-Cache-up-to-4_20-GHz

[4] C. Cusack, C. Martens and P. Mutreja, "Volunteer computing using casual games," in *Proceedings of Future Play 2006 International Conference on the Future of Game Design and Technology*, London, Canada, 2006, pp. 1–8.

[5] D. P. Anderson, J. Cobb, E. Korpela, M. Lebofsky and D. Werthimer, "SETI@ home: an experiment in public-resource computing," *Commun ACM*, vol. 45, pp. 56–61, 2002.

[6] D. P. Anderson, "Volunteer computing: the ultimate cloud." *ACM Crossroads*, vol. 16, pp. 7–10, 2010.

[7] S. Tiezzi. (2015, Aug. 4). U.S. to challenge China for world's fastest supercomputer. *The Diplomat* [Online]. Available: http://thediplomat.com/2015/08/us-to-challenge-china-for-worlds-fastest-supercomputer/

[8] A. Borji and L. Itti, "Human vs. computer in scene and object recognition," in *2014 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, , 2014, pp. 113–120.

[9] V. D. Silva, D. Kroening and G. Weissenbacher, "A survey of automated techniques for formal software verification," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 27, pp. 1165–1178, 2008.

[10] C. A. Rohl, C. E. Strauss, K. M. Misura and D. Baker, "Protein structure prediction using Rosetta," *Meth. Enzymol.*, vol. 383, pp. 66–93, 2004.

[11] U. Tellioglu, G. G. Xie, J. P. Rohrer and C. Prince, "Whale of a crowd: Quantifying the effectiveness of crowd-sourced serious games," in *Computer Games: AI, Animation, Mobile, Multimedia, Educational and Serious Games (CGAMES), 2014*, 2014, pp. 1–7.

[12] D. Dean, S. Gaurino, L. Eusebi, A. Keplinger, T. Pavlik, R. Watro, A. Cammarata, J. Murray, K. McLaughlin and J. Cheng, "Lessons learned in game development for crowdsourced software formal verification," presented at *2015 USENIX Summit on Gaming, Games, and Gamification in Security Education (3GSE 15)*, Washington, D.C., 2015.

[13] V. Curtis, "Motivation to Participate in an Online Citizen Science Game: A Study of Foldit," *Science Communication,* October 16, 2015.

[14] *The Science Behind Foldit.* (2016, Jan. 14). Foldit. [Online]. Available: http://fold.it/portal/info/about#whygame

[15] M. J. Coren. (2011, Sep. 20). Foldit gamers solve riddle of HIV enzyme in 3 weeks. *Scientific American* [Online]. Available: http://www.scientificamerican.com/article/Foldit-gamers-solve-riddle/

[16] U. Tellioglu. Quantifying the effectiveness of crowd-sourced serious games. M.S. thesis, Dept. Info. Sci., Naval Postgraduate School, Monterey, CA, 2014.

[17] *GIMPS history.* (2016, Jan 07). Great Internet Mersenne Prime Search. [Online]. Available: http://www.mersenne.org/various/history.php

[18] K. Chang. (2016. January 21). New biggest prime number = 2 to the 74 mil ... uh, it's big. *New York Times* [Online]. Available: http://www.nytimes.com/2016/01/22/science/new-biggest-prime-number-mersenne-primes.html

[19] C. Franzoni and H. Sauermann, "Crowd science: The organization of scientific research in open collaborative projects," *Research Policy,* vol. 43, pp. 1–20, 2014.

[20] M. Yuen, I. King and K. Leung, "A survey of crowdsourcing systems," in *2011 IEEE Third International Conference on Privacy, Security, Risk and Trust (PASSAT) and 2011 IEEE Third International Conference on Social Computing (SocialCom),* , 2011, pp. 766–773.

[21] J. Howe, "The rise of crowdsourcing," *Wired Magazine,* vol. 14, pp. 1–4, 2006.

[22] D. C. Brabham, "The myth of amateur crowds: A critical discourse analysis of crowdsourcing coverage," *Information, Communication & Society,* vol. 15, pp. 394–410, 2012.

[23] *Twitch is 4th in peak U.S. Internet traffic*. (2014. Feb 05). Twitch. [Online]. Available: http://blog.twitch.tv/2014/02/twitch-community-4th-in-peak-us-Internet-traffic/.

[24] M. McWhertor. (2014, February 14). *How Twitch is crowd-sourcing an amazing Pokémon multiplayer game* [Online]. Available: http://www.polygon.com/2014/2/14/5411790/twitch-plays-pokemon-creator-interview-twitchplayspokemon.

[25] *TPP victory! the thundershock heard around the world*. (2014. Mar 1). Twitch. [Online]. Available: http://blog.twitch.tv/2014/03/twitch-prevails-at-pokemon/.

[26] L. von Ahn, "Games with a purpose," *Computer,* vol. 39, pp. 92–94, 2006.

[27] L. von Ahn and L. Dabbish, "Labeling images with a computer game," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems,* 2004, pp. 319–326.

[28] N. Yee, "Motivations for play in online games," *CyberPsychology & Behavior,* vol. 9, pp. 772–775, 2006.

[29] L. Achterbosch, R. Pierce and G. Simmons, "Massively Multiplayer Online Role-playing Games: The Past, Present, and Future," *Comput.Entertain.*, vol. 5, pp. 9:1-9:33, March 2008.

[30] U. Tellioglu, B. Kartal, M. Simsek and O. Eryilmaz, "Call of duty: Can Turkey benefit from crowd-sourced serious games to strengthen its cyber security capabilities?" in *Proceedings of 7th Int. Conf. on Information Security and Cryptology (ISCTurkey),* Istanbul, Turkey 2014, pp. 117–122.

[31] S. Mavandadi, S. Dimitrov, S. Feng, F. Yu, U. Sikora, O. Yaglidere, S. Padmanabhan, K. Nielsen and A. Ozcan, "Distributed medical image analysis and diagnosis through crowd-sourced games: a malaria case study," *PloS One,* vol. 7, pp. e37245, 2012.

[32] G. Klein, K. Elphinstone, G. Heiser, J. Andronick, D. Cock, P. Derrin, D. Elkaduwe, K. Engelhardt, R. Kolanski and M. Norrish, "seL4: Formal verification of an OS kernel," in *Proceedings of the ACM SIGOPS 22nd Symposium on Operating Systems Principles,* Big Sky, Montana, 2009, pp. 207–220.

[33] A. Keplinger, M. Barry, J. N. Rushton, G. Izzo and Q. Zheng, "Solution 1: Circuitbot and dynamakr," in *Lessons Learned in Game Development for Crowdsourced Software Formal Verification*. 2015 USENIX Summit on Gaming, Games, and Gamification in Security Education (3GSE 15), Washington, D.C., 2015, pp. 3–5.

[34] T. Pavlik, C. Conner, J. Burke, M. Burns, W. Dietl, S. Cooper, M. Ernst and Z. Popovic, "Solution 2: Flow jam and paradox," in *Lessons Learned in Game Development for Crowdsourced Software Formal Verification.* 2015 USENIX Summit on Gaming, Games, and Gamification in Security Education (3GSE 15) , Washington, D.C., 2015, pp. 6–8.

[35] R. Watro, K. Moffitt, J. Ostwald, E. Church, D. Wyschogrod, A. Lapets and L. Kennard, "Solution 3: Ghost map and hyperspace," in *Lessons Learned in Game Development for Crowdsourced Software Formal Verification*. 2015 USENIX Summit on Gaming, Games, and Gamification in Security Education (3GSE 15), Washington, D.C., 2015, pp. 9–10.

[36] R. Watro, K. Moffitt, T. Hussain, D. Wyschogrod, J. Ostwald, D. Kong, C. Bowers, E. Church, J. Guttman and Q. Wang, "Ghost Map: Proving software correctness using games," *Securware 2014,* pp. 223, 2014.

[37] A. Cammarata and A. Tomb, "Solution 4: Stormbound and monster proof," in *Lessons Learned in Game Development for Crowdsourced Software Formal Verification*. 2015 USENIX Summit on Gaming, Games, and Gamification in Security Education (3GSE 15), Washington, D.C., 2015, pp. 11–13.

[38] J. Murray, H. Logas and J. Whitehead, "Solution 5: Xylem and binary fission," in *Lessons Learned in Game Development for Crowdsourced Software Formal Verification*. 2015 USENIX Summit on Gaming, Games, and Gamification in Security Education (3GSE 15), Washington, D.C., pp. 14–16.

[39] M. Yilmaz. Framework for evaluating loop invariant detection games in relation to automated dynamic invariant detectors. M.S. thesis, Dept. Info. Sci., Naval Postgraduate School, Monterey, CA, 2015.

[40] S. Cooper, *A Framework for Scientific Discovery through Video Games.* New York: Morgan & Claypool, 2014.

[41] *What is Rosetta@home?* (2009, Apr 8). Rosetta@home. [Online]. Available: https://boinc.bakerlab.org/rosetta/rah_about.php.

[42] *Quick guide to Rosetta and its graphics*. (2007, Nov. 10). Rosetta@home. [Online]. Available: https://boinc.bakerlab.org/rah_graphics.php.

[43] R. Das, B. Qian, S. Raman, R. Vernon, J. Thompson, P. Bradley, S. Khare, M. D. Tyka, D. Bhat, D. Chivian, D. E. Kim, W. H. Sheffler, L. Malmström, A. M. Wollacott, C. Wang, I. Andre and D. Baker, "Structure prediction for CASP7 targets using extensive all-atom refinement with Rosetta@home," *Proteins: Structure, Function, and Bioinformatics*, vol. 69, pp. 118–128, 2007.

[44] S. Rogers, *Level Up! the Guide to Great Video Game Design.* New York: John Wiley & Sons, 2014.

[45] *Soloist Hall of Fame* . (2016, February 12). Foldit. [Online]. Available: http://fold.it/portal/players/s_all.

[46] S. Cooper, A. Treuille, J. Barbero, A. Leaver-Fay, K. Tuite, F. Khatib, A. C. Snyder, M. Beenen, D. Salesin, D. Baker and Z. Popovic, "The challenge of

designing scientific discovery games," in *Proceedings of the Fifth International Conference on the Foundations of Digital Games,* Monterey, California, 2010, pp. 40–47.

[47] F. Khatib, S. Cooper, M. D. Tyka, K. Xu, I. Makedon, Z. Popovic, D. Baker and F. Players, "Algorithm discovery by protein folding game players," *Proc. Natl. Acad. Sci. U. S. A.*, vol. 108, pp. 18949-18953, November 22, 2011.

[48] F. Khatib, F. DiMaio, S. Cooper, M. Kazmierczyk, M. Gilski, S. Krzywda, H. Zabranska, I. Pichova, J. Thompson and Z. Popović, "Crystal structure of a monomeric retroviral protease solved by protein folding game players," *Nature Structural & Molecular Biology,* vol. 18, pp. 1175–1177, 2011.

[49] J. Barone, C. Bayer, R. Copley, N. Barlow, M. Burns, S. Rao, G. Seelig, Z. Popovic, S. Cooper and N. Players, "Nanocrafter: Design and evaluation of a DNA nanotechnology game," presented at Proceedings of the 10th International Conference on the Foundations of Digital Games, Pacific Grove, CA,2015.

[50] D. Cook. (2007, May 15). The circle of life: an analysis of the game product life cycle. *Gamasutra* [Online]. Available: http://www.gamasutra.com/view/feature/1453/the_circle_of_life_an_analysis_of_.php?print=1

[51] D. Shaymall. (2014, May 27). A comprehensive list of metrics for free to play games. *Gamesbrief* [Online]. Available: http://www.gamesbrief.com/2014/05/a-comprehensive-list-of-metrics-for-free-to-play-games/.

[52] T. V. Fields, "Game industry metrics terminology and analytics case study," in *Game Analytics,* M. Seif El-Nasr, A. Drachen, A. Canossa, Eds. London, UK: Springer, 2013, pp. 53–71.

[53] N. Lovell. (2011, Nov 16). Whales, dolphins and minnows – the beating heart of a free-to-play game. *Gamesbrief* [Online]. Available: http://www.gamesbrief.com/2011/11/whales-dolphins-and-minnows-the-beating-heart-of-a-free-to-play-game/

[54] S. Carmichael. (2013, Mar 14). What it means to be a 'whale' — and why social gamers are just gamers. *Venturebeat* [Online]. Available: http://venturebeat.com/2013/03/14/whales-and-why-social-gamers-are-just-gamers/.

[55] *1157b: 60 residue monomer design*. (2015, Nov. 19). Foldit. [Online]. Available: https://fold.it/portal/node/2001453

[56] *Foldit* (n.d.). Drupal Showcase. [Online]. Available: http://www.drupalshowcase.com/drupal-showcase/foldit. Accessed Mar. 7, 2016.

[57] *Score* (n.d.). Foldit Wiki. [Online]Available: http://foldit.wikia.com/wiki/Score. Accessed Mar 20, 2016.

[58] S. Cooper, F. Khatib, I. Makedon, H. Lu, J. Barbero, D. Baker, J. Fogarty, Popovic Zoran and F. players, "Analysis of social gameplay macros in the foldit cookbook," in *Proceedings of the 6th International Conference on Foundations of Digital Games*, Bordeaux, France, 2011, pp. 9–14.

[59] A. Boyle. (2011. Sept 20). *Gamers solve molecular puzzle that baffled scientists* [Online]. Available: http://www.nbcnews.com/science/science-news/gamers-solve-molecular-puzzle-baffled-scientists-f6C10402813

[60] NPR Staff. (2011. Oct 3.). *When scientists fail, it's time to call in the gamers.* [Online]. Available: http://www.npr.org/2011/10/02/140979241/when-scientists-fail-its-time-to-call-in-the-gamers

[61] M. Moskovitch. (2011. Sept 20). *Online game Foldit helps anti-Aids drug quest* [Online]. Available: http://www.bbc.com/news/technology-14986013

[62] *FoldIt - Awards and Honors* (2013. Sept 25). Center for Games Science [Online]. Available: http://centerforgamescience.org/blog/portfolio/foldit/

[63] C. Norman, "2011 International Science & Engineering Visualization Challenge," *Science*, vol. 335, pp. 525, 2012.

[64] Electronic Entertainment Design and Research. Deconstructing mobile & table gaming 2015. EEDAR. 2015 [Online]. Available: http://www.eedar.com/reports/EEDAR_Deconstructing_Mobile_and_Tablet_Gaming_2015.pdf

[65] H. Sauermann and C. Franzoni, "Crowd science user contribution patterns and their implications," *Proc. Natl. Acad. Sci. U. S. A.*, vol. 112, pp. 679–684, Jan 20, 2015.

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center
   Ft. Belvoir, Virginia

2. Dudley Knox Library
   Naval Postgraduate School
   Monterey, California